Modern Information Retrieval

Chapter 8

Text Classification

Introduction
A Characterization of Text Classification
Unsupervised Algorithms
Supervised Algorithms
Feature Selection or Dimensionality Reduction
Evaluation Metrics
Organizing the Classes - Taxonomies

Introduction

- Since ancient times, librarians had to deal with the issue of storing documents for later retrieval and reading
- As time passed, the size of the collection grew and the problem hardened
- Searching hundreds of books sequentially, looking for a particular book of interest, became impractical
- To alleviate the problem, librarians started **labeling the documents**

Introduction

- A very first approach for labeling documents was to assign an unique identifier to each document
- However, this not solve the more generic problem of finding documents on a **specific subject or topic**
- In this case, the natural solution is to
  - group documents by common topics, and
  - name these groups with meaningful labels
- Each labeled group is call a **class**
- This type of classification task is commonly referred to as **topic classification**

Introduction

- Classes can be used to describe other characteristics of the documents, such as
  - Language, genre, quality, authority, popularity, spam nature
- The process of inserting the documents into the classes is commonly referred to as **text classification**
- Sometimes, classes are referred to as **categories** and the classification process is called **text categorization**
- We consider that classification and categorization are the same process

Introduction

- A related problem is to separate a set of documents into subsets, without labeling them
- Since each subset has no label, we do not say that we have a class
- Instead, we refer to each subset as a **cluster** and the separation process as a **text clustering** procedure
- In here, we will treat clustering as a simpler variant of the classification problem

Introduction

- Text classification provides a means to organize information
- To illustrate, consider the case of a large engineering company that executes dozens of large projects yearly
- As a result, thousands of documents related to the business of the company are produced
- These documents constitute a valuable asset to support the business decision making process
- That is, text classification is a key technology for knowledge organization in modern enterprises

A Characterization of Text Classification

Machine Learning

Machine Learning

- Machine Learning: design of algorithms that **learn** patterns existent in data provided as input
- The patterns learned are then used to make predictions relative to unseen and new data
- Machine learning algorithms are fundamentally dependent on a learning phase
- Depending on the learning mechanism used, the machine learning algorithm can be of three types:
  - Supervised learning
  - Unsupervised learning
  - Semi-supervised learning

Machine Learning

- **Supervised learning** requires learning a function from **training data** provided as input
- The training data is composed of document-class pairs indicating proper classes for given documents
- This training data is then used to learn a classification function

Machine Learning

- **Unsupervised learning:** no training data is provided
- Example of algorithms: neural network models, independent component analysis, and clustering
- For text classification purposes, clustering is the type of unsupervised learning algorithm of most interest
- **Semi-supervised learning** combines a small amount of labeled data with a larger amount of unlabeled data

A Characterization of Text Classification

The Text Classification Problem

The Text Classification Problem

- The text classification problem can be formally stated as follows
  - Let a collection $\mathcal{D}$ of documents, and a set $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ of L classes with their respective labels
  - A text classifier is a binary function $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow \{0, 1\}$
- That is, $\mathcal{F}$ is a function that assigns a value of 0 or 1 to each pair $[d_j, c_p]$, such that $d_j \in \mathcal{D}$ and $c_p \in \mathcal{C}$
- If the value assigned is 1, we say that the document d_j is a member of class c_p
- If the value assigned is 0, we say that the document d_j is not a member of class c_p

The Text Classification Problem

- This definition is broad and admits both supervised and unsupervised text classification algorithms
- In general, however, to obtain text classification of high accuracy one has to adopt a supervised algorithm
- Types of classification algorithms:
  - **multi-label**: two or more classes might be assigned to a single document
  - **single-label**: the classifier assigns a single class to each document

The Text Classification Problem

- The definition of our classification function $\mathcal{F}$ is binary
- However, the function $\mathcal{F}$ might be built to compute a degree of membership of document d_j in the class c_p
- Then, we say that there are a number of documents that are candidates to be members of class c_p
- Further, we can present them to a user ordered by the decreasing values of $\mathcal{F}(d_j, c_p)$

A Characterization of Text Classification

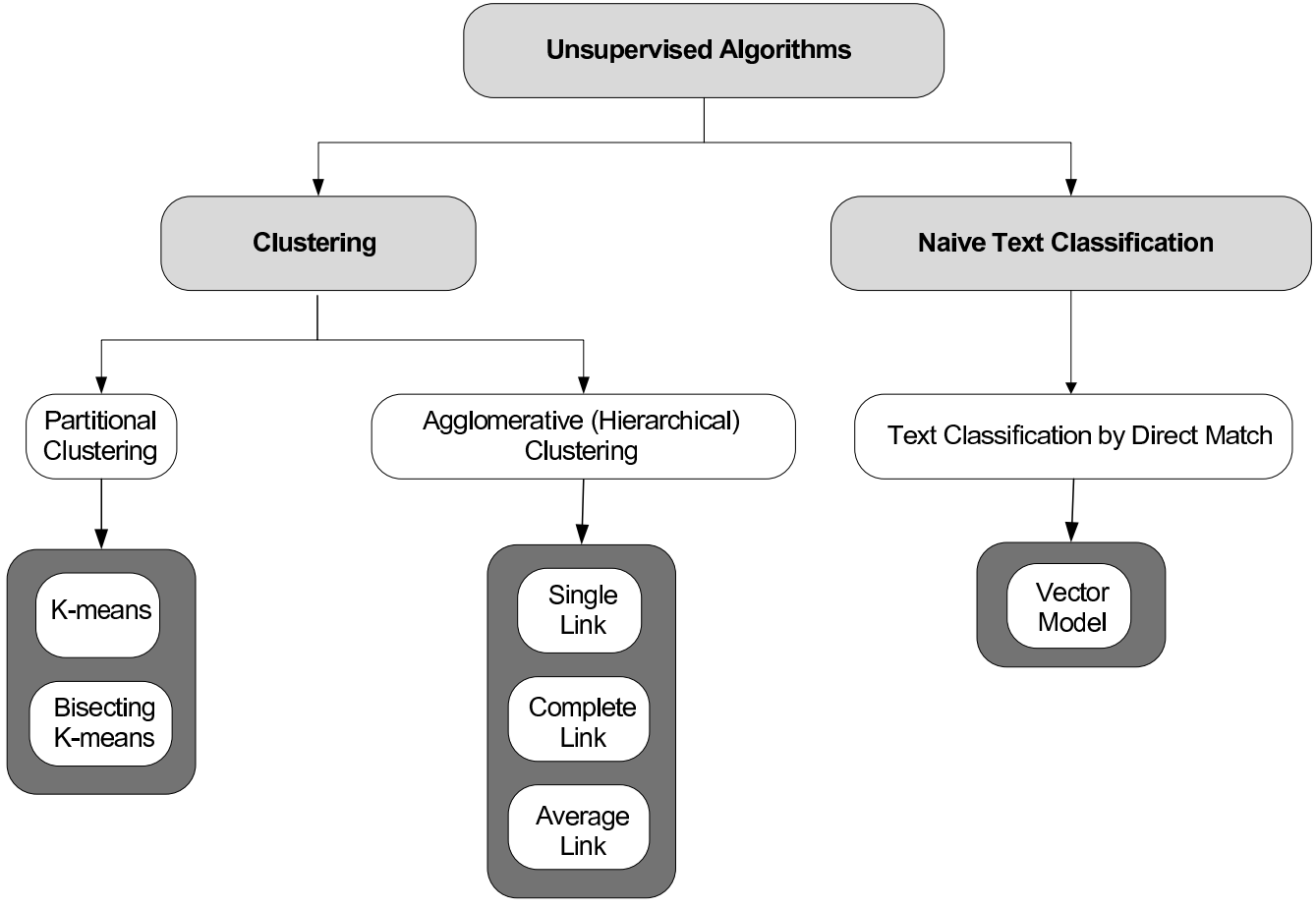
Text Classification Algorithms

Text Classification Algorithms

- Text classification algorithms are either **unsupervised** or **supervised**
- The first ones are suited for large collections for which no training data is available
- The second ones lead to better results, but require the availability of training data

Text Classification Algorithms

- Figure below illustrates the unsupervised algorithms we discuss here



Text Classification Algorithms

- An algorithm is said to be **supervised** when it uses information provided by humans as input data
- In the standard case, a set of classes and examples of documents for each class are provided
- The examples are determined by human specialists and constitute the **training set**
- The training set is used to **learn** a classification function

Text Classification Algorithms

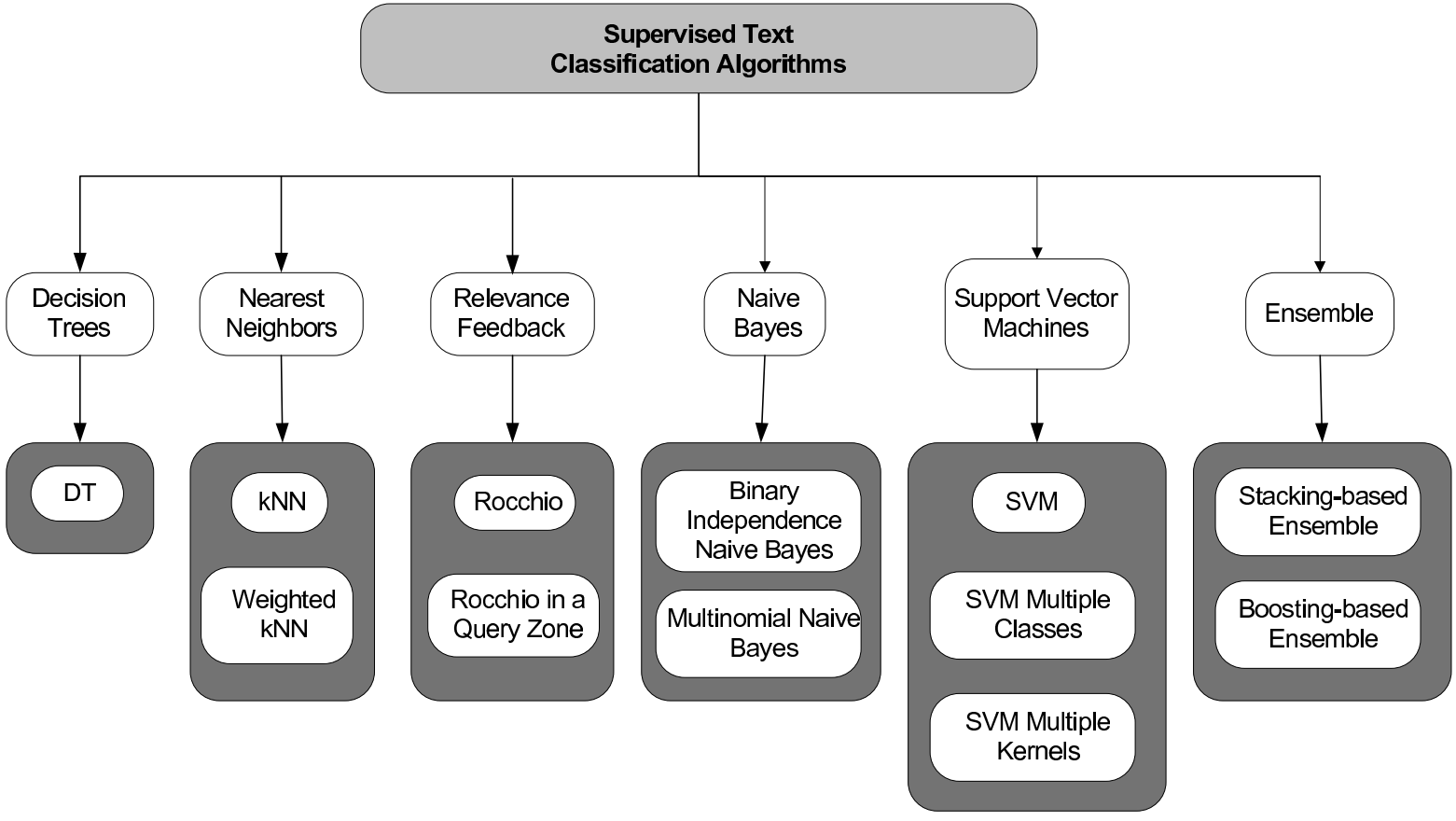
- For instance, a human expert can specify 4 classes with an average number of 100 documents in each of them
- The classifier can then determine, for instance, the terms that occur most frequently in each class
- These terms can be used to produce an initial description of that class
- An analogous process can be repeated in the case the examples are provided for clusters

Text Classification Algorithms

- The larger the number of training examples, usually the better is the fine tuning of the classifier
- However, care must be exercised to avoid that the classifier becomes specific to the training examples
- In this case, the classifier cannot be used to predict the classes of new and unseen objects
- Such an event is commonly referred to as **overfitting**
- To evaluate the classifier, we apply it to a set of unseen objects whose classes have been determined a priori
- This set of objects is commonly referred to as **test set**

Text Classification Algorithms

- Figure below illustrates the supervised text classification algorithms we discuss here



Unsupervised Algorithms

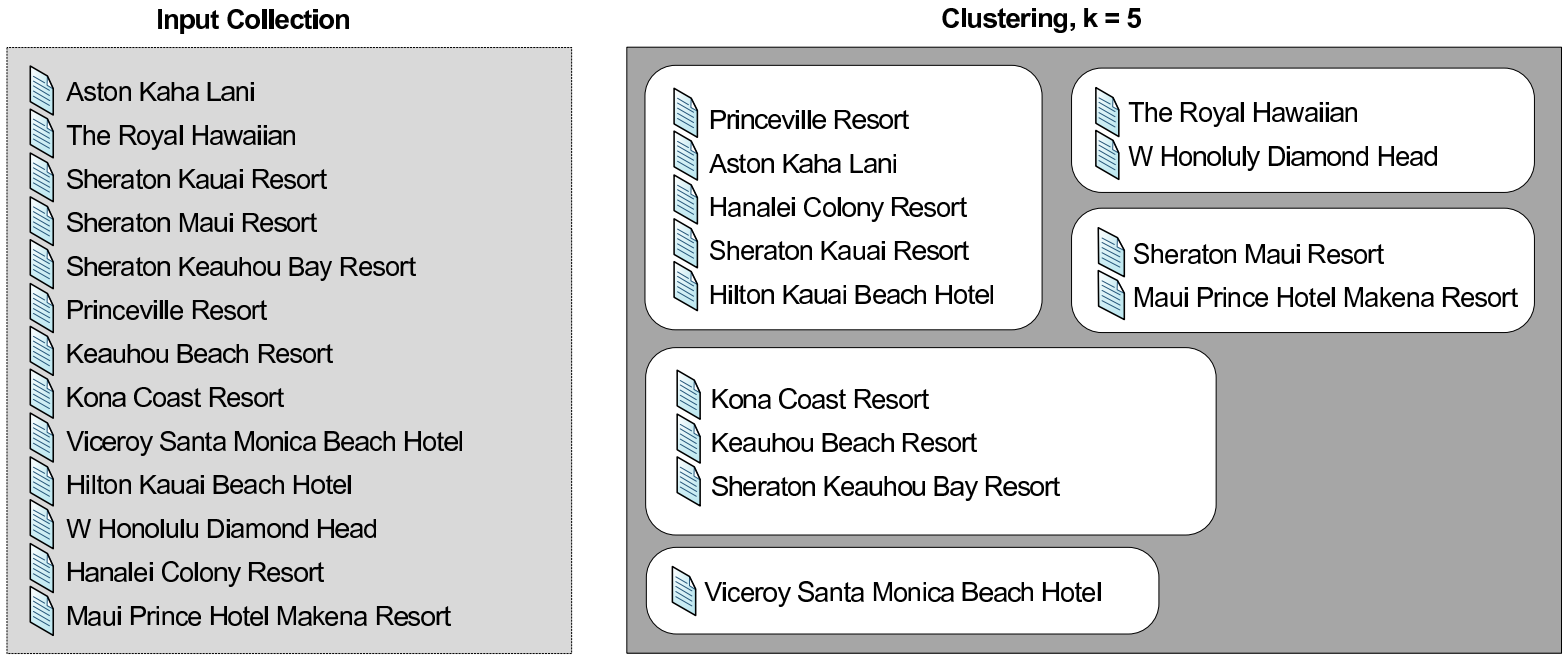
Clustering

Clustering

- Assume that the only input data are the documents and that not even class labels are provided
- In this case, the task of the classifier is to separate the documents in groups in fully automatic fashion
- This procedure is frequently referred to as **clustering**

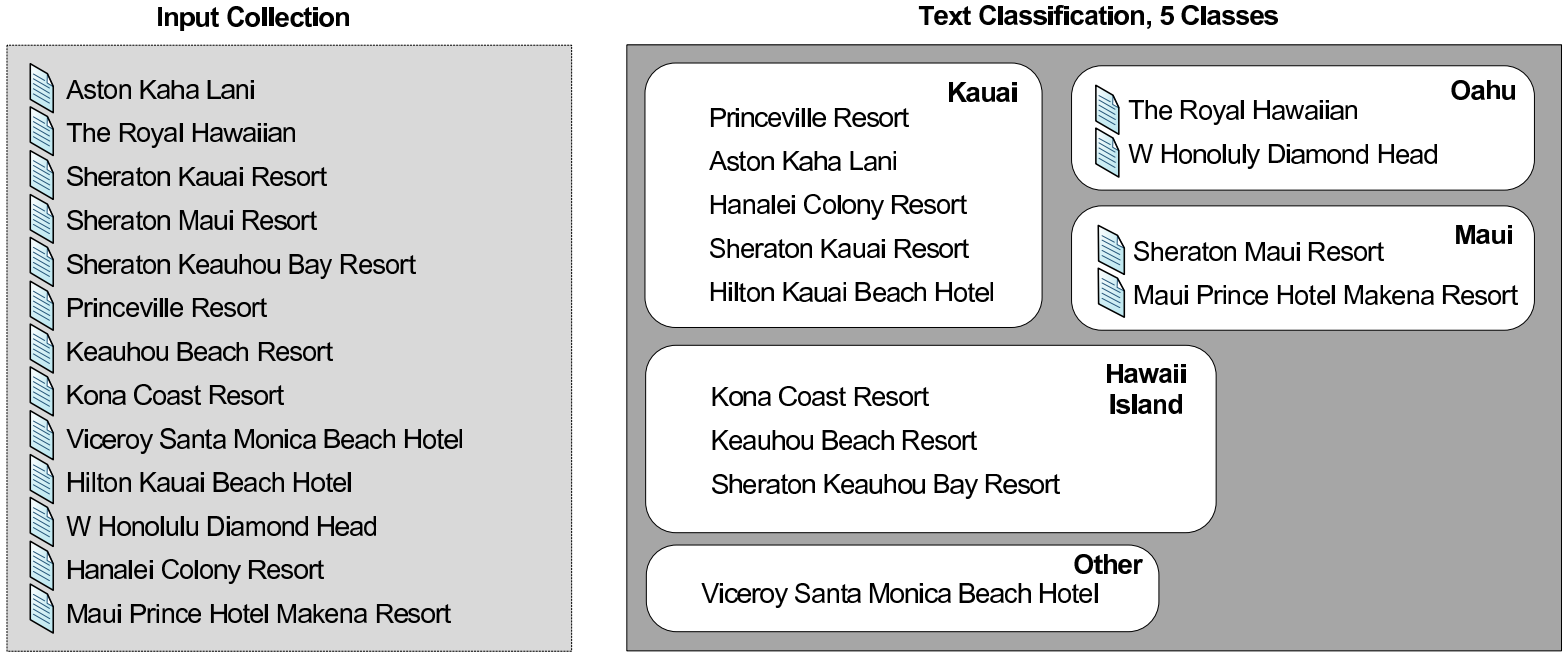
Clustering

- Unsupervised text classification applied to Web pages of hotels in Hawaii: **clustering**



Clustering

- Unsupervised text classification applied to Web pages of hotels in Hawaii: **assignment of classes to hotels**
- Each class is composed of hotels located in a same island



Clustering

- While this clustering is naturally appealing to humans, it is difficult to be generated by an automatic procedure
- The reason is that the hotel Web pages contain many terms in common
- Without human understanding, it is very hard to establish that terms describe the hotel locations
- Thus, most likely, an automatic clustering algorithm will not generate the clusters shown

Clustering

- Even if the right clusters had been produced, we would still have to label these clusters
- Labels generated automatically tend to be very distinct from labels specified by humans
- Thus, solving the whole classification problem with no provision of any form of human knowledge is hard
- This continues to be an excessively complex task, particularly because of the labelling of the clusters
- Unsupervised classification is more effective when class labels have been provided

Unsupervised Algorithms

K-means Clustering

K-means Clustering

- In K-Means clustering, the number K of clusters to be generated is provided as input
- Each cluster is represented by a center point, such as a **centroid**
- The docs are partitioned among the K clusters, with each doc assigned to the cluster with closest centroid
- Once this is done, centroids are recomputed and the whole process is repeated until centroids to not change

K-means Clustering

- The **basic K-means** method operates in **batch mode**
  - That is, all documents are classified before recomputing centroids

- Let each document d_j be represented as vector $\vec{d_j}$, given by

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

where

  - $w_{i,j}$ is the weight of term k_i in document d_j
  - t is the size of the vocabulary

K-means Clustering

- K-means clustering works as follows:
  1. **Initial step.** Select K docs randomly and use them as the initial centroids $\vec{\triangle}_p$ in the document space

$$\vec{\triangle}_p = \vec{d}_j$$

  2. **Assignment Step.** Assign each of the N documents of the collection to the cluster with closest centroid
     - Distance is the inverse of the similarity, which is computed by using the cosine formula of the vector model

$$sim(d_j, c_p) = \frac{\vec{\triangle}_p \bullet \vec{d}_j}{|\vec{\triangle}_p| \times |\vec{d}_j|}$$

K-means Clustering

3. **Update Step.** Recompute or adjust the centroids for each cluster c_p

$$\vec{\triangle}_p = \frac{1}{size(c_p)} \sum_{\vec{d}_j \in c_p} \vec{d}_j$$

4. **Final Step.** Repeat the **assignment** and **update** steps until no centroid changes

K-means Clustering

- The second version of K-means operates **online**
  - In this case, centroids are recomputed after individual documents are classified
- It works as follows:
  1. **Initial Step.** Select K documents randomly and use them as the initial centroids
  2. **Assignment Step.** For each document d_j repeat
     - assign document d_j to the cluster with closest centroid
     - recompute the centroid of that cluster to include d_j
  3. **Final Step.** Repeat assignment step until no centroid changes.

K-means Clustering

- It is argued that, for text document collections, online K-means works better than batch K-means
- K-means clustering may work well in some situations and not so well in others
- The choice on the number K of clusters, for instance, is a critical step in the algorithm

Unsupervised Algorithms

Bisecting K-means

Bisecting K-means

- This algorithm builds a hierarchy of clusters that branches into two clusters at each step
- This is done by applying K-means repeatedly, with K=2, which works as follows

Bisecting K-Means

1. **Initial Step:** assign all documents to a single cluster.
2. **Split Step:** apply K-means, with $K = 2$, to the largest cluster.
3. **Selection Step:** if a stop criteria (e.g., no cluster is larger than a pre-defined size) is satisfied
   - then stop execution
   - otherwise, select the cluster with the largest number of documents and go back to the Split Step

Unsupervised Algorithms

Hierarchical Agglomerative Clustering

Hierarchical Clustering

- Hierarchical clustering methods attempt to create hierarchies of clusters
- They operate by either
  - decomposing a large cluster into smaller ones, or by
  - agglomerating previously defined clusters into larger ones

Hierarchical Clustering

- The general functioning of a hierarchical document clustering algorithm can be described as follows

1. Receive as input a set of N documents to be clustered and a $N \times N$ similarity (distance) matrix
2. Assign each document to its own cluster, such that N clusters are produced, each cluster containing one document
3. Find the closest pair of clusters and merge them into a single cluster, reducing the number of clusters to $N - 1$
4. Recompute the distances between the new cluster and each of the old clusters
5. Repeat steps 3 and 4 until all documents are clustered into a single cluster of size N

Hierarchical Clustering

- The distances between the clusters is the similarities between the respective documents within the clusters
- Step 4 introduces the notion of similarity or distance between two clusters
- The way these **cluster distances** are computed defines three variants of the algorithm
- These variantes are known respectively as *single-link*, *complete-link*, and *average-link* clustering

Hierarchical Clustering

- **Single-Link Algorithm:** the cluster distance is equal to the shortest similarity between any document of one cluster to any document of the other cluster
- **Complete-Link Algorithm:** the cluster distance is equal to the lowest similarity between any document of one cluster to any document of the other cluster
- **Average-Link Algorithm:** the cluster distance is equal to the average similarity between all documents of one cluster and all documents of the other cluster

Unsupervised Algorithms

Naive Text Classification

Naive Text Classification

- A second form of unsupervised classification is to specify the classes without any information on training examples
- **Naive Classification:**
  - Given a collection $\mathcal{D}$ of documents, a set $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ of L classes with their respective labels
  - Determine a method to automatically associate one or more classes of $\mathcal{C}$ with each document of the collection

Naive Text Classification

- A simple classification algorithm can be implemented by directly matching document terms to class labels
- The coverage of the classification might be improved by defining alternative labels for each class
- These labels are often referred to as **synonyms**
- Partial matches can be quantified

Naive Text Classification

Text Classification by Direct Match

1. Receive as input a collection $\mathcal{D}$ of documents and a set $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ of L classes with their respective labels.
2. Represent the documents and classes by weighted term vectors.
   - The class vectors are built using the terms of the class labels
3. For each document $d_j \in \mathcal{D}$ do
   - retrieve the classes $c_p \in \mathcal{C}$ whose labels contain terms that match terms in d_j
   - for each pair $[d_j, c_p]$, compute a vector based ranking given by

$$sim(d_j, c_p) = \frac{\vec{d_j} \bullet \vec{c_p}}{|\vec{d_j}| \times |\vec{c_p}|}$$

   - associate with document d_j the classes c_p with the highest values of $sim(d_j, c_p)$

Naive Text Classification

- Naive text classification might yield good results for vertical collections
- This is particularly the case when a taxonomy is available

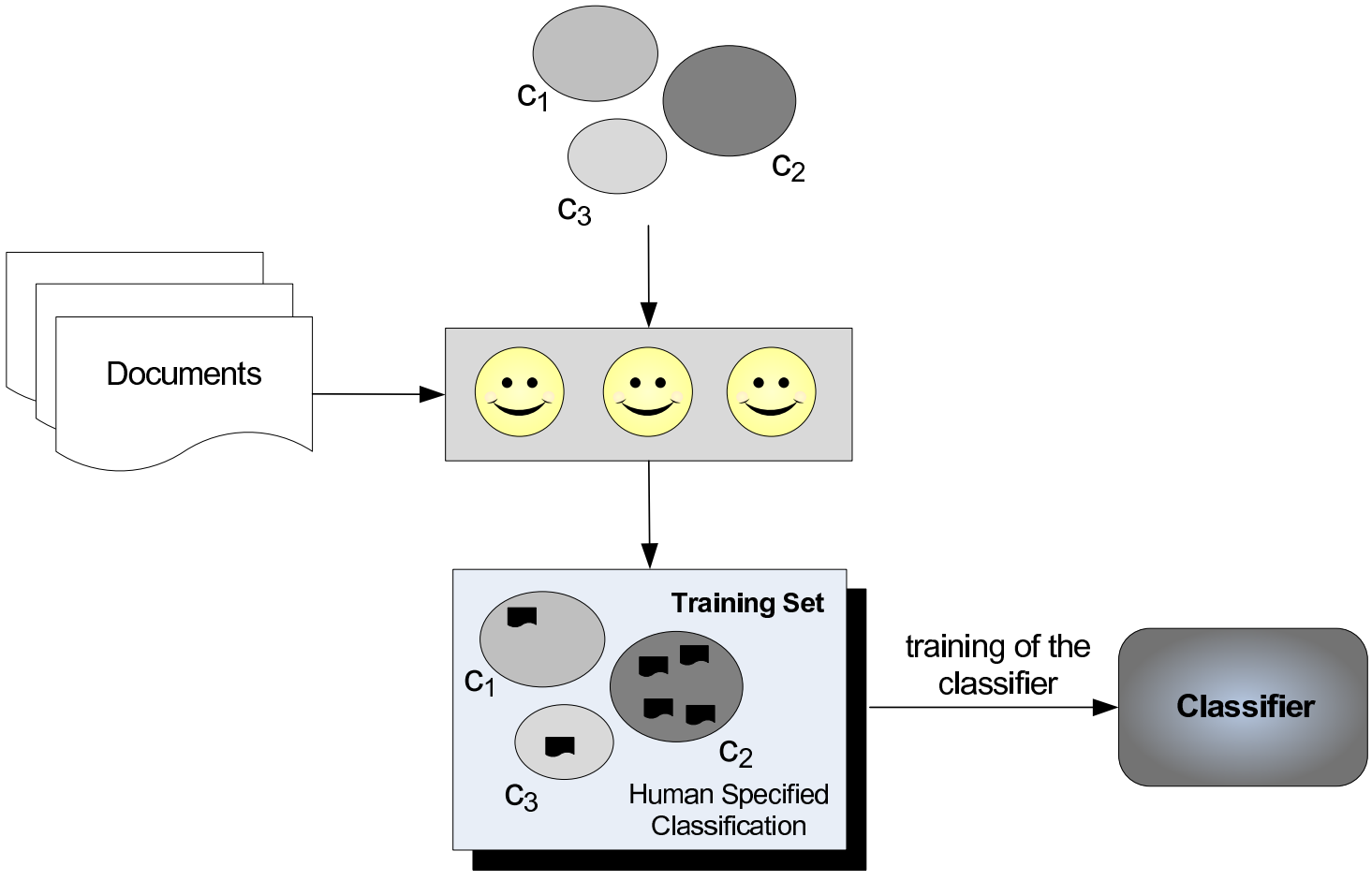
Supervised Algorithms

Supervised Algorithms

- The text classification problem can be made more sophisticated by using a **training set**
- A training set is a set of the input examples of documents pre-classified by humans:
  - Given a sub-collection $\mathcal{D}_t \subset \mathcal{D}$ of training documents
  - A training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow \{0, 1\}$ assigns a value of 0 or 1 to each pair $[d_j, c_p]$, $d_j \in \mathcal{D}_t$ and $c_p \in \mathcal{C}$, according to the judgement of human specialists
- The training set function $\mathcal{T}$ is used to fine tune the classifier

Supervised Algorithms

- The training phase of a classifier

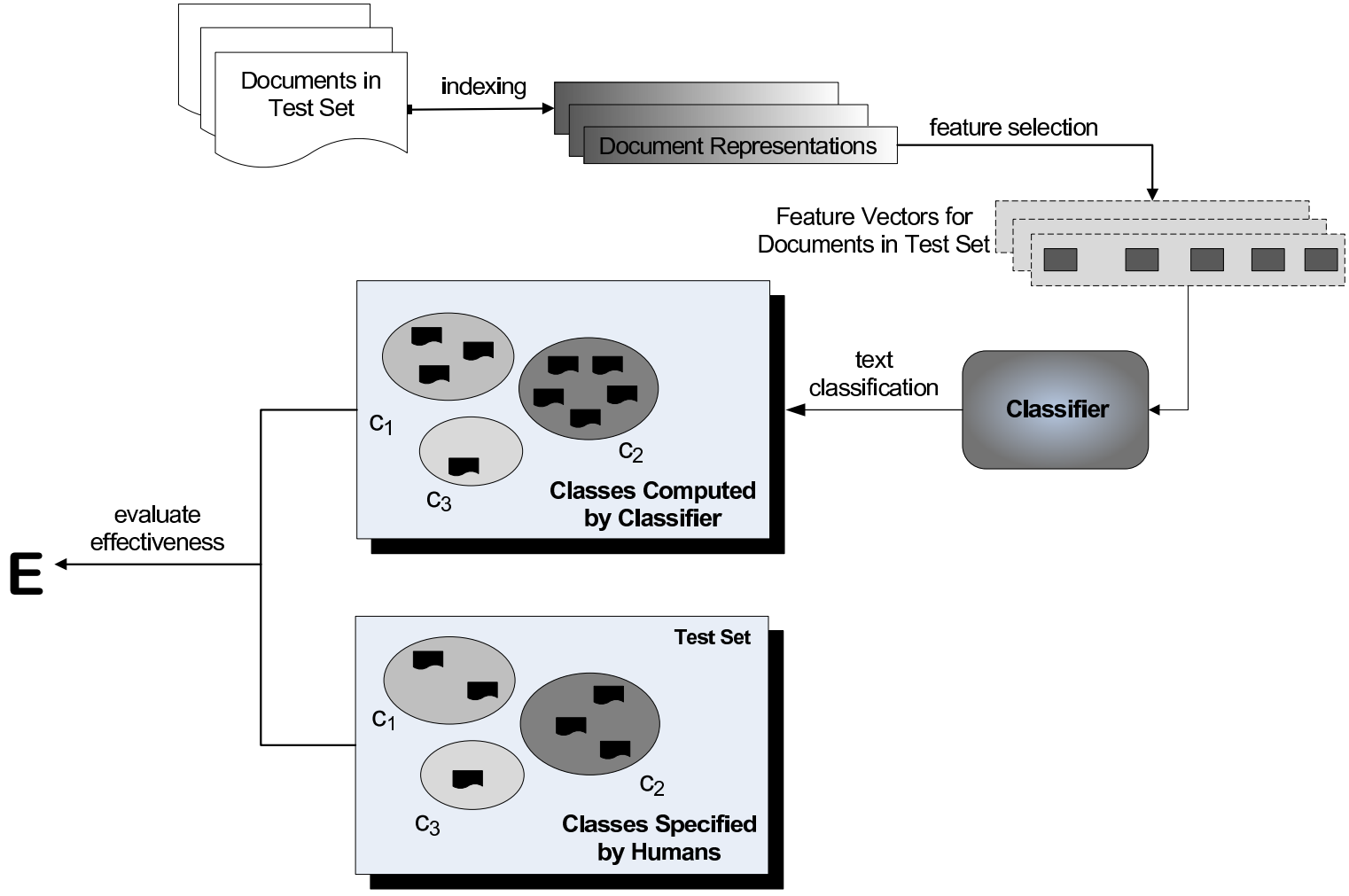


Supervised Algorithms

- To evaluate the classifier, we use a set of documents for which the correct classes are known - the **test set**
- As with the training set, the assignment of classes to documents in the test set is done by human specialists
- The test set is used to evaluate the classifier in a two step process:
  - use the classifier to assign classes to the documents in the test set, and
  - compare the classes assigned by the classifier with those specified by the human specialists

Supervised Algorithms

- Classification process and evaluation of a supervised classifier



Supervised Algorithms

- Once the classifier has been trained and validated, it can be used to classify new and unseen documents
- If the classifier is operating properly, it classifies new documents into the proper classes with high effectiveness

Supervised Algorithms

Decision Trees

Decision Trees

- A **decision tree (DT)** is a supervised classification method
- It uses a training set to build classification rules organized as paths in a tree
- These tree paths can then be used to classify documents outside the training set
- One of the advantages of the approach is that the rules in the tree are amenable to human interpretation
- This data structure facilitates the interpretation of the results of the classification process

Basic Technique

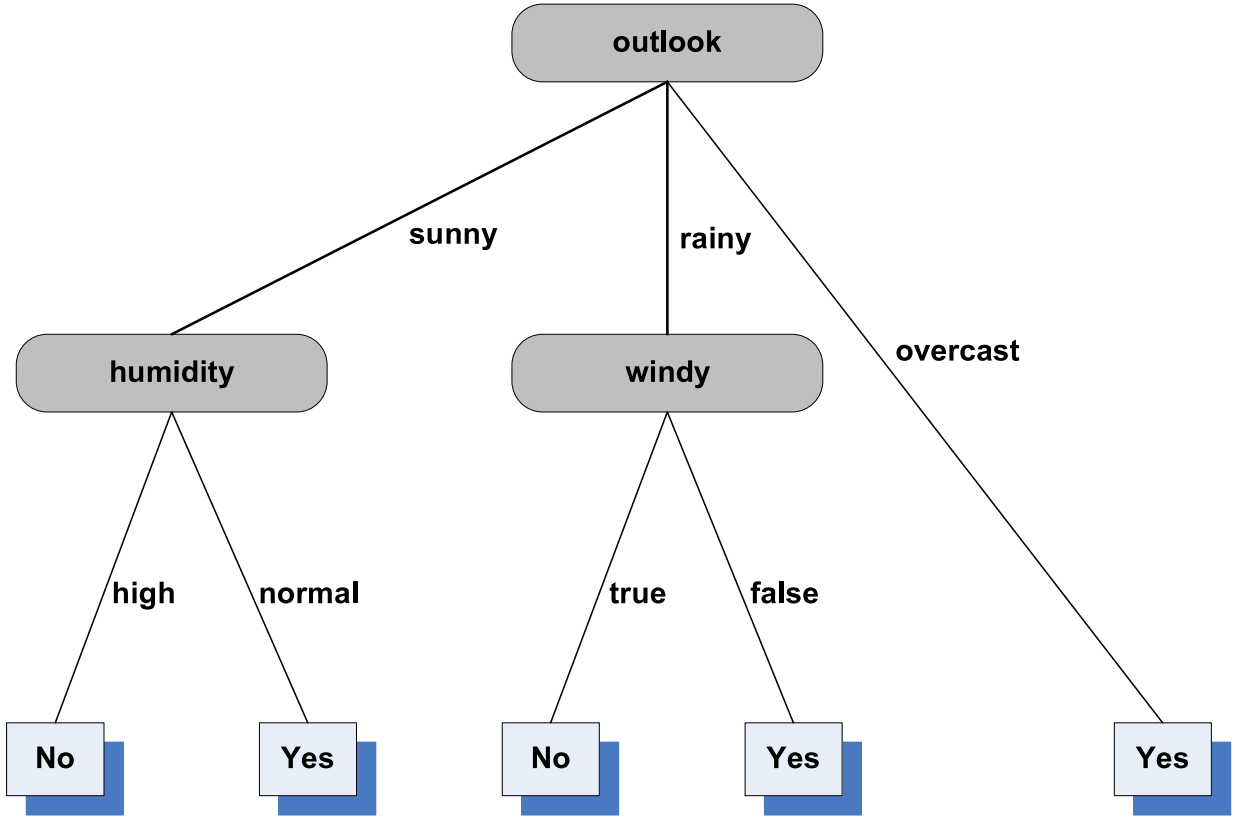
- Consider the small relational database in Table below

	Id	Play	Outlook	Temperature	Humidity	Windy
Training set	1	yes	rainy	cool	normal	false
	2	no	rainy	cool	normal	true
	3	yes	overcast	hot	high	false
	4	no	sunny	mild	high	false
	5	yes	rainy	cool	normal	false
	6	yes	sunny	cool	normal	false
	7	yes	rainy	cool	normal	false
	8	yes	sunny	hot	normal	false
	9	yes	overcast	mild	high	true
	10	no	sunny	mild	high	true
Test Instance	11	?	sunny	cool	high	false

- A decision tree for this database is a data structure that allows predicting the values of a given attribute

Basic Technique

- To illustrate, the DT below allows predicting the values of the attribute Play, given that we know the values for attributes like Outlook, Humidity, and Windy



Basic Technique

- The internal nodes are associated with attribute names and the edges are associated with attribute values
- A recursive traversal of the DT allows deciding the most appropriate value for the attribute “Play”.
- In the case of tuple 11, the decision would be “not to play” based on the rule induced by the path
$(Outlook = sunny) \wedge (Humidity = high)$

	Id	Play	Outlook	Temperature	Humidity	Windy
Test Instance	11	?	sunny	cool	high	false

Basic Technique

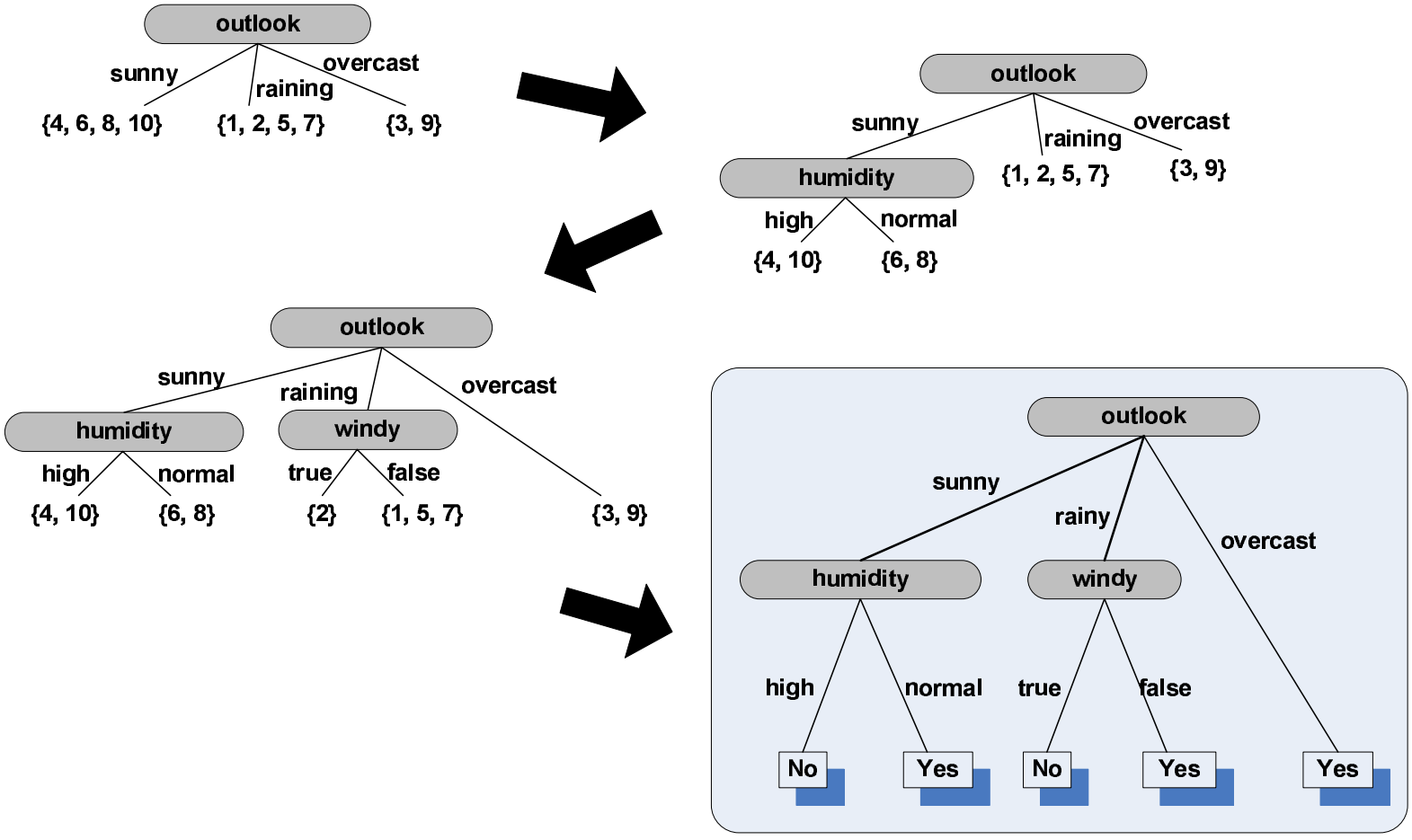
- Notice that our predictions are based on the instances that we have seen in the example database
- A new instance that violates these predictions will lead to an erroneous prediction
- We say that the example database works as a training set for building the decision tree

The Splitting Process

- Given a training set, a DT model for the database can be built using a recursive splitting strategy
- Consider that the objective is to build a decision tree for predicting the values of attribute Play
- The first step is to select one of the attributes, other than Play, to be the root of the decision tree
- The corresponding attribute values are then used to split the tuples in the database into subsets
- For each subset of tuples, a second splitting attribute is selected and the process is repeated

The Splitting Process

- Figure below shows an step by step example of this splitting process



The Splitting Process

- Notice that the splitting process is strongly affected by the order with which the split attributes are selected
- Depending of this ordering, the tree might become unbalanced
- This is one of the key challenges while devising a splitting strategy
- Usually, balanced or near-balanced trees work better and more efficient for predicting attribute values
- Thus, a common rule of thumb is to select attributes that reduce the average path length of the leaves

Classification of Documents

- For document classification, with each internal node in the tree we associate an index term
- With each leave in the tree we associate a document class
- Further, with the edges we associate binary predicates that indicate the presence/absence of an index term

Classification of Documents

- Let V be a set of nodes
- A tree $T = (V, E, r)$ is an acyclic graph on V where
  - $E \subseteq V \times V$ is the set of edges and
  - $r \in V$ is called the root of T
- Given an edge (v_i, v_j), v_i is considered the father node and v_j is the child node
- We designate by I the set of all internal nodes and by $\overline{I}$ the set of all leaf nodes

Classification of Documents

- Given a tree T, we can associate information on documents and their classes with the tree
- By doing so, we create a decision tree for document classification, as follows
- Let
  - $K = \{k_1, k_2, \ldots, k_t\}$ be the set of index terms of a doc collection
  - C be the set of all classes, as before
  - P be a set of logical predicates on the index terms

Classification of Documents

- Further, let
  - A decision tree $DT = (V, E; r; l_I, l_L, l_E)$ is a six-tuple where $(V; E; r)$ is a tree whose root is r
  - $l_I : I \rightarrow K$ is a function that associates with each internal node of the tree one or more index terms
  - $l_L : \overline{I} \rightarrow C$ is a function that associates with each non-internal (leaf) node a class $c_p \in C$
  - $l_E : E \rightarrow P$ is a function that associates with each edge of the tree a logical predicate from P

Classification of Documents

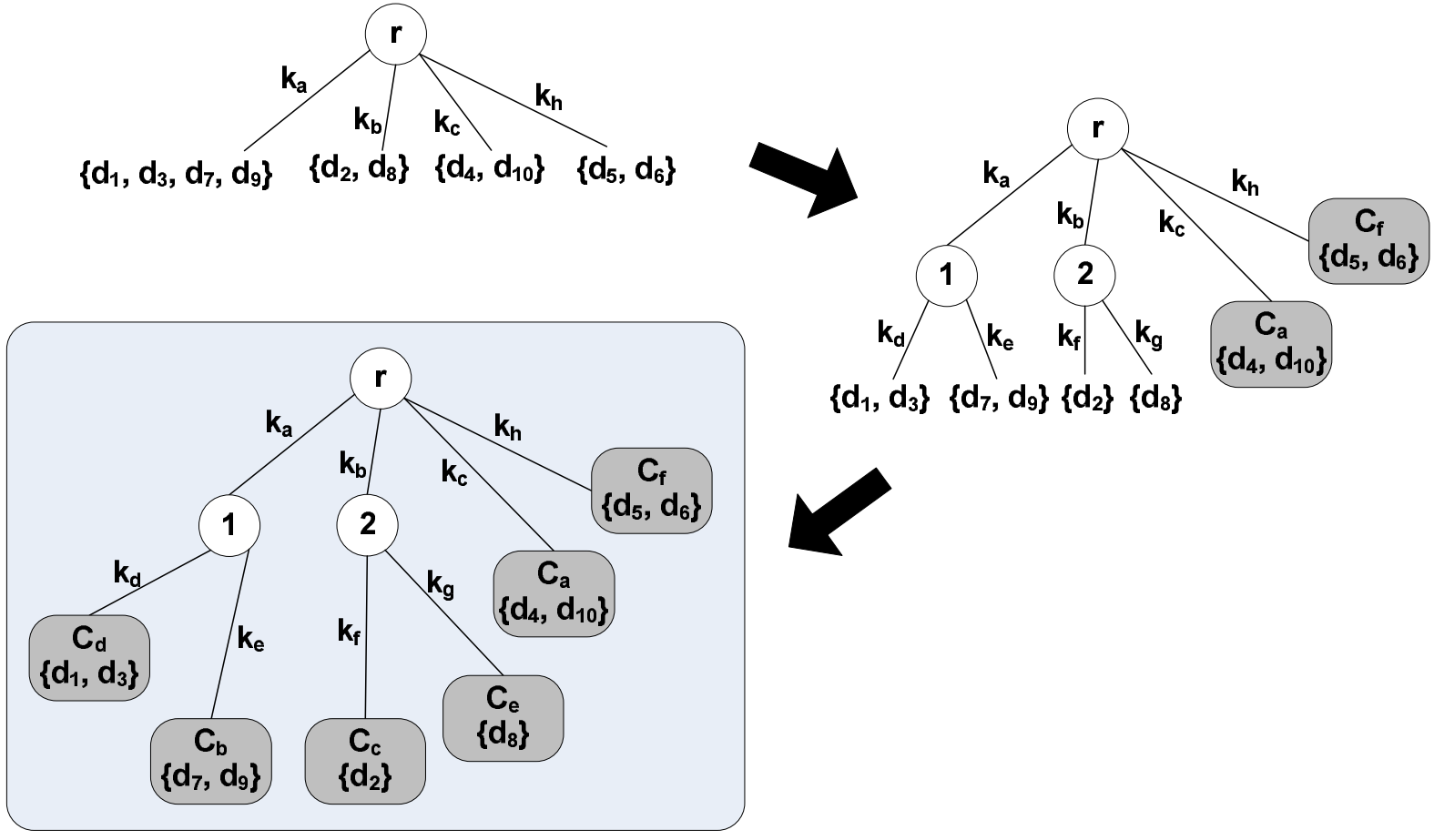
- Given a training set, a decision tree model for a class c_p can be built using a recursive splitting strategy
- The **first step** is to associate all documents to the root
- The **second step** is to select a number of index terms that provide a good separation of the documents
- To illustrate, consider that terms k_a, k_b, k_c, and k_h have been selected for this first split
- Then, the documents in the root are separated into 4 subsets

Classification of Documents

- The edge connecting a subset to the root is labelled with the respective index term
- Notice that a same document might appear in more than one subset
- Following, new splitting terms are selected for each doc subset and the process is repeated recursively
- At each branch, the recursion is stopped whenever all documents of a subset belong to a same class

Classification of Documents

- Splitting process for inducing a decision tree for a collection of documents



Classification of Documents

- The crucial point of this process is the procedure for selecting splitting terms
- While distinct procedures can be used, information gain and entropy are the most commonly used ones
- Selection of terms with high information gain tends
  - to increase the number of branches at a given level, and
  - to reduce the number of documents in each resultant subset
- This tends to yield smaller and less complex decision trees

Classification of Documents

- Decision trees have some inherent problems such as missing or unknown values
- These appear when the document to be classified does not contain some terms used to build the DT
- In this case, it is not clear which branch of the tree should be traversed
- To avoid this problem, we can delay the construction of the tree until a new document is presented for classification
- The tree is then built based on the features presented in the document, therefore avoiding the problem

Classification Algorithms

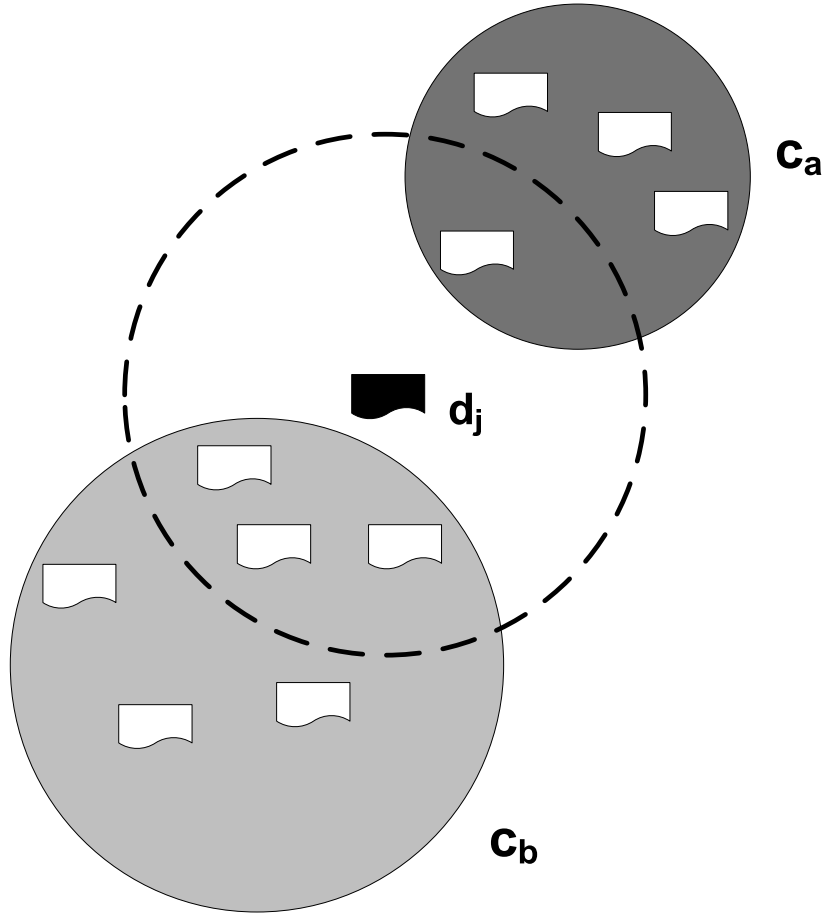
The kNN Classifier

The kNN Classifier

- The kNN (k-nearest neighbor) classifier is a type of on-demand (or lazy) classifier
- **Lazy classifiers** do not build a classification model a priori
- The classification is performed at the moment a new document d_j is given to the classifier
- It is based on the classes of the k nearest neighbors of d_j, computed using a distance function
- This is accomplished as follows:
  - determine the k nearest neighbors of d_j in a training set
  - use the classes of these neighbors to determine a class for d_j

The kNN Classifier

- An example of a 4-NN classification process



Classification of Documents

- In the kNN algorithm, to each document-class pair $[d_j, c_p]$ we assign a score S_{d_j,c_p}, given by:

$$S_{d_j,c_p} = \sum_{d_t \in N_k(d_j)} similarity(d_j, d_t) \times \mathcal{T}(d_t, c_p)$$

  where

  - $N_k(d_j)$ is the set of the k nearest neighbors of d_j in the training set
  - $\mathcal{T}(d_t, c_p)$, the training set function, returns 1 if d_t belongs to class c_p, and 0 otherwise

- The classifier assigns to document d_j the class(es) c_p with the highest score(s)

Classification of Documents

- The cosine measure between the two document vectors is commonly used as the similarity function
- One problem with kNN is performance
- The classifier has to compute distances between the document to be classified and *all* training documents
- Another issue is how to choose the “best” value for k

Classification Algorithms

The Rocchio Classifier

The Rocchio Classifier

- The Rocchio relevance feedback process allows modifying an user query based on user feedback
- The motivation is to produce a new query that better approximates the interest of the user
- The Rochio formula can be adapted for text classification
- The idea is to interpret the training set as feedback information

Basic Technique

- The Rocchio relevance feedback process is based on the classic vector model
- It considers that each document d_j is represented as a weighted term vector $\vec{d_j}$, given by

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

where $w_{i,j}$ is the weight of term k_i in document d_j and t is the size of the vocabulary

Basic Technique

- In this case,
  - terms that belong to training docs of a given class c_p are said to provide positive feedback
  - terms that belong to training docs outside class c_p are said to provide negative feedback
- All feedback information can be summarized by a centroid vector in the term space
- A new test document can be classified by measuring its distance to the centroid

Classification of Documents

- A Rochio classifier for a class c_p is represented as a weighted vector

$$\vec{c}_p = \{w_{1,p}, w_{2,p}, \ldots, w_{t,p}\}$$

where t is the total number of index terms in the collection, as before

Classification of Documents

- Let
  - n_p be the number of documents in class c_p according to the training set
  - N_t be the total number of documents in the training set
- Then, the vector for each class c_p is obtained by computing a centroid for that class as follows

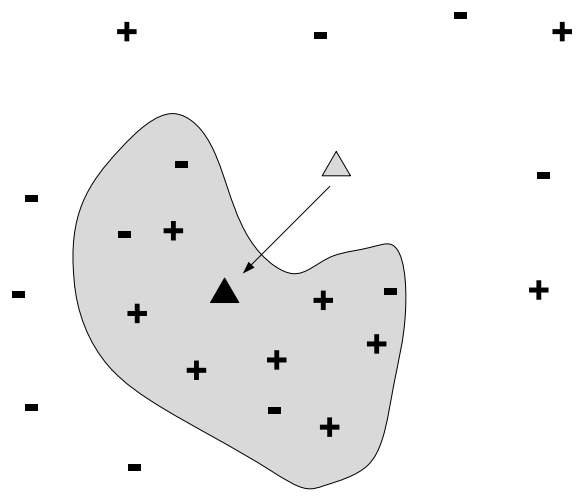
$$\vec{c}_p = \frac{\beta}{n_p} \sum_{d_j \in c_p} \vec{d}_j - \frac{\gamma}{N_t - n_p} \sum_{d_l \notin c_p} \vec{d}_l \qquad (1)$$

Classification of Documents

- Notice that
  - terms of training documents that belong to class c_p receive positive weights
  - terms of training documents outside class c_p receive negative weights

Classification of Documents

- Rocchio text classification represented in a space of document terms



- Plus signals indicate terms that belong to training documents in a given class c_p
- Minus signals indicate terms that belong to training documents outside class c_p

Classifying New Documents

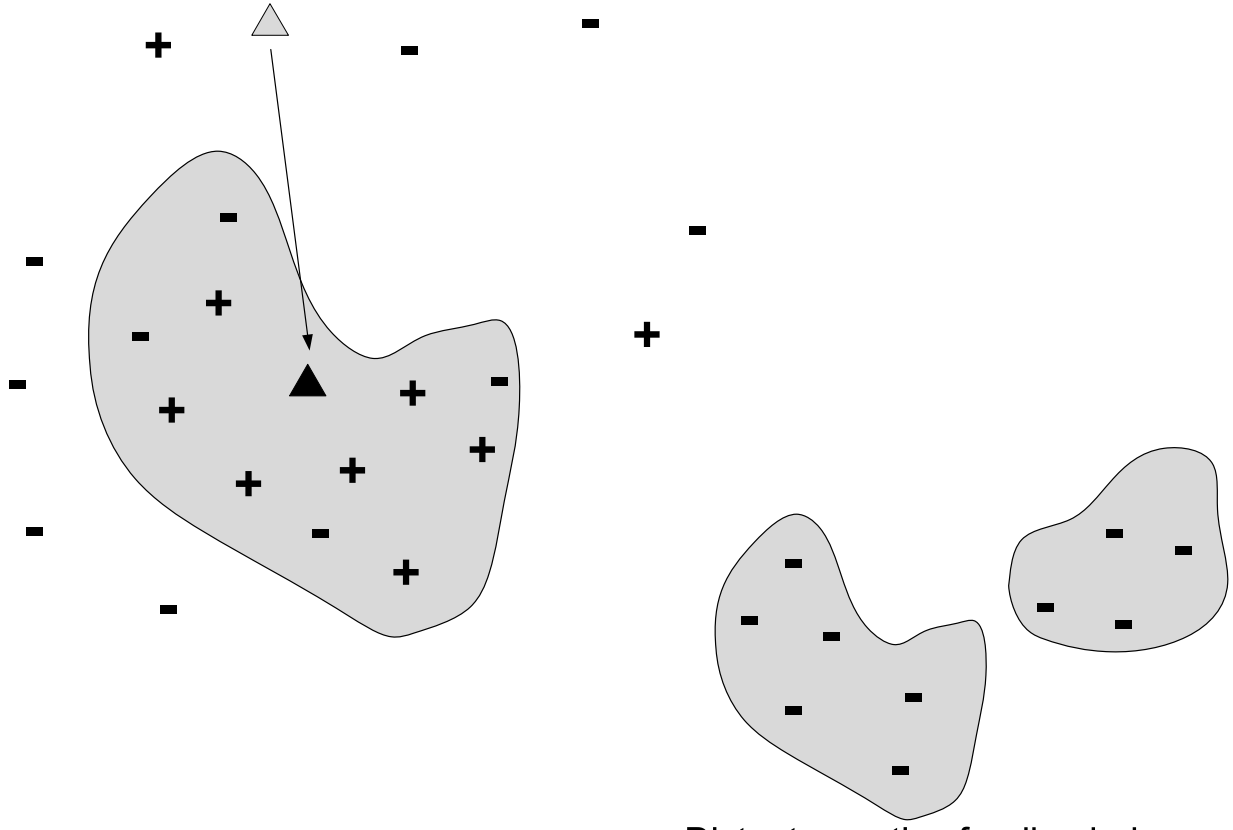
- The Rocchio classifier assigns to each document-class pair $[d_j, c_p]$ a score $S(d_j, c_p)$

$$S(d_j, c_p) = |\vec{c}_p - \vec{d}_j|$$

- Classes with the highest scores $S(d_j, c_p)$ are assigned to document d_j

Rocchio in a Query Zone

- For specific domains, negative feedback might move the centroid away from the topic of interest



Distant negative feedback documents might shift the centroid away from the positive terms.

Rocchio in a Query Zone

- To reduce this effect, we can decrease the number of documents used for negative feedback
- We can use only the **most positive** documents among all the documents that provide negative feedback
- These documents are usually referred to as **near-positive documents**

Rocchio in a Query Zone

- Near-positive documents are selected as follows
  - Let $\vec{c}_{p+}$ be the centroid of the training documents that belong to class c_p, i.e., the positive documents
  - Take all training documents outside c_p and measure their distances to $\vec{c}_{p+}$
  - Those that have the smaller distances to the centroid are taken as the near-positive documents

Supervised Algorithms

Probabilistic Naive Bayes Document Classification

Naive Bayes

- Probabilistic classifiers
  - Assign to each document-class pair a probability that the document belongs to that class
  - Given a document d_j, they assign to each document-class pair $[d_j, c_p]$ the probability $P(c_p|\vec{d}_j)$

Naive Bayes Classifier

- To compute the probability $P(c_p|\vec{d}_j)$, a probabilistic classifier applies the Bayes theorem, as follows:

$$P(c_p|\vec{d}_j) = \frac{P(c_p) \times P(\vec{d}_j|c_p)}{P(\vec{d}_j)}$$

where

- $P(\vec{d}_j)$ is the probability that a random selection of a document returns a document represented by $\vec{d}_j$
- $P(c_p)$ is the probability that a random selection of a document returns a document in class c_p

Naive Bayes Classifier

- The computation of $P(\vec{d}_j|c_p)$ requires simplifications to be done efficiently
- The most common simplification is to assume independence among the index terms
- Because this assumption, classifiers that are based on it are called **Naive Bayes classifiers**

Naive Bayes Classifier

- There are many variants of Naive Bayes classifiers
- Of these, one of the best known is based on the classic probabilistic model
- A doc d_j is represented by a vector of binary weights indicating presence or absence of index terms:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

where $w_{i,j} = 1$ if term k_i occurs in document d_j and $w_{i,j} = 0$, otherwise

Naive Bayes Classifier

- With each document-class pair $[d_j, c_p]$, the classifier assigns a score $S(d_j, c_p)$ given by the ratio

$$S(d_j, c_p) = \frac{P(c_p|\vec{d}_j)}{P(\overline{c}_p|\vec{d}_j)}$$

where

  - $P(c_p|\vec{d}_j)$ is the probability that document d_j belongs to class c_p
  - $P(\overline{c}_p|\vec{d}_j)$ is the probability that document d_j does not belong to class c_p

- Clearly, $P(c_p|\vec{d}_j) + P(\overline{c}_p|\vec{d}_j) = 1$

Naive Bayes Classifier

- Applying Bayes, we obtain

$$S(d_j, c_p) \sim \frac{P(\vec{d}_j|c_p)}{P(\vec{d}_j|\overline{c}_p)}$$

- To estimate these probabilities, we adopt the independence assumption, that is

$$P(\vec{d}_j|c_p) = \prod_{k_i \in \vec{d}_j} P(k_i|c_p) \times \prod_{k_i \notin \vec{d}_j} P(\overline{k}_i|c_p)$$

- Also,

$$P(\vec{d}_j|\overline{c}_p) = \prod_{k_i \in \vec{d}_j} P(k_i|\overline{c}_p) \times \prod_{k_i \notin \vec{d}_j} P(\overline{k}_i|\overline{c}_p)$$

Naive Bayes Classifier

- We can deduce the equation for the score $S(d_j, c_p)$, as follows

$$
\begin{aligned}
S(d_j, c_p) &\sim \sum_{k_i} w_{i,j} \left(\log \frac{p_{iP}}{1 - p_{iP}} + \log \frac{1 - q_{iP}}{q_{iP}} \right) \\
p_{iP} &= P(k_i | c_p) \\
q_{iP} &= P(k_i | \overline{c}_p)
\end{aligned}
$$

where

- p_{iP}: probability that the ith index term is present in a doc randomly selected from class c_p
- q_{iP}: probability that the ith index term is present in a doc randomly selected from outside class c_p

Naive Bayes Classifier

- p_{iP} and q_{iP} can be estimated from the set $\mathcal{D}_t$ of training documents, as follows

$$p_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(c_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)} = \frac{1 + n_{i,p}}{2 + n_p}$$

$$q_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(\overline{c}_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(\overline{c}_p|d_j)} = \frac{1 + (n_i - n_{i,p})}{2 + (N_t - n_p)}$$

where

- $P(c_p|d_j) \in \{0, 1\}$ and $P(\overline{c}_p|d_j) \in \{0, 1\}$ are given by the training set
- the $n_{i,p}$, n_i, n_p, and N_t are as defined in probabilistic model

Multinomial Naive Bayes Classifier

- The Naive Bayes classifier above assumes that term weights are binary
- A variant of this model is to take into account that a same term occurs multiple times in a document
- To classify a document d_j in a class c_p we use:

$$P(c_p|\vec{d}_j) = \frac{P(c_p) \times P(\vec{d}_j|c_p)}{P(\vec{d}_j)}$$

where

- $P(\vec{d}_j)$ is the prior document probability
- $P(c_p)$ is the prior class probability

Multinomial Naive Bayes Classifier

- The prior class probability is given by

$$P(c_p) = \frac{\sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)}{N_t} = \frac{n_p}{N_t}$$

where $P(c_p|d_j) \in \{0, 1\}$ is obtained directly from the training set of size N_t

Multinomial Naive Bayes Classifier

- The prior document probability is given by

$$P(\vec{d}_j) = \sum_{p=1}^{L} P_{prior}(\vec{d}_j|c_p) \times P(c_p)$$

where L is the total number of classes, and

$$P_{prior}(\vec{d}_j|c_p) = \prod_{k_i \in \vec{d}_j} P(k_i|c_p) \times \prod_{k_i \notin \vec{d}_j} [1 - P(k_i|c_p)]$$

and

$$P(k_i|c_p) = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(c_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)} = \frac{1 + n_{i,p}}{2 + n_p}$$

Multinomial Naive Bayes Classifier

- Notice that these equations do not take into account term frequencies
- To compute the key probabilities $P(\vec{d_j}|c_p)$, we modify our formulation to include term frequencies, as follows
  - Consider that the terms of a document d_j of class c_p are drawn from a known distribution
  - Each single term draw is interpreted as a Bernoulli trial in which the probability of success is given by $P(k_i|c_p)$
  - Further, each term k_i is drawn as many times as its document frequency $f_{i,j}$

Multinomial Naive Bayes Classifier

- In this case, terms in a document follow a multinomial probabilistic distribution, that is,

$$P(\vec{d_j}|c_p) = F_i! * \prod_{i=1}^{t} \frac{P(k_i|c_p)^{f_{i,j}}}{f_{i,j}!}$$

- where t is the size of the vocabulary and

$$F_i = \sum_{i=1}^{t} f_{i,j}$$

- Notice that F_i provides a measure of the document length

Multinomial Naive Bayes Classifier

- The term probabilities can be estimated from the training set, as follows:

$$P(k_i|c_p) = \frac{\sum_{d_j \in \mathcal{D}_t} f_{i,j} P(c_p|d_j)}{\sum_{\forall k_i} \sum_{d_j \in \mathcal{D}_t} f_{i,j} P(c_p|d_j)}$$

  where $\mathcal{D}_t$ is the set of training documents

- Further, $P(c_p|d_j) \in \{0, 1\}$ is obtained directly from the training set

Classification Algorithms

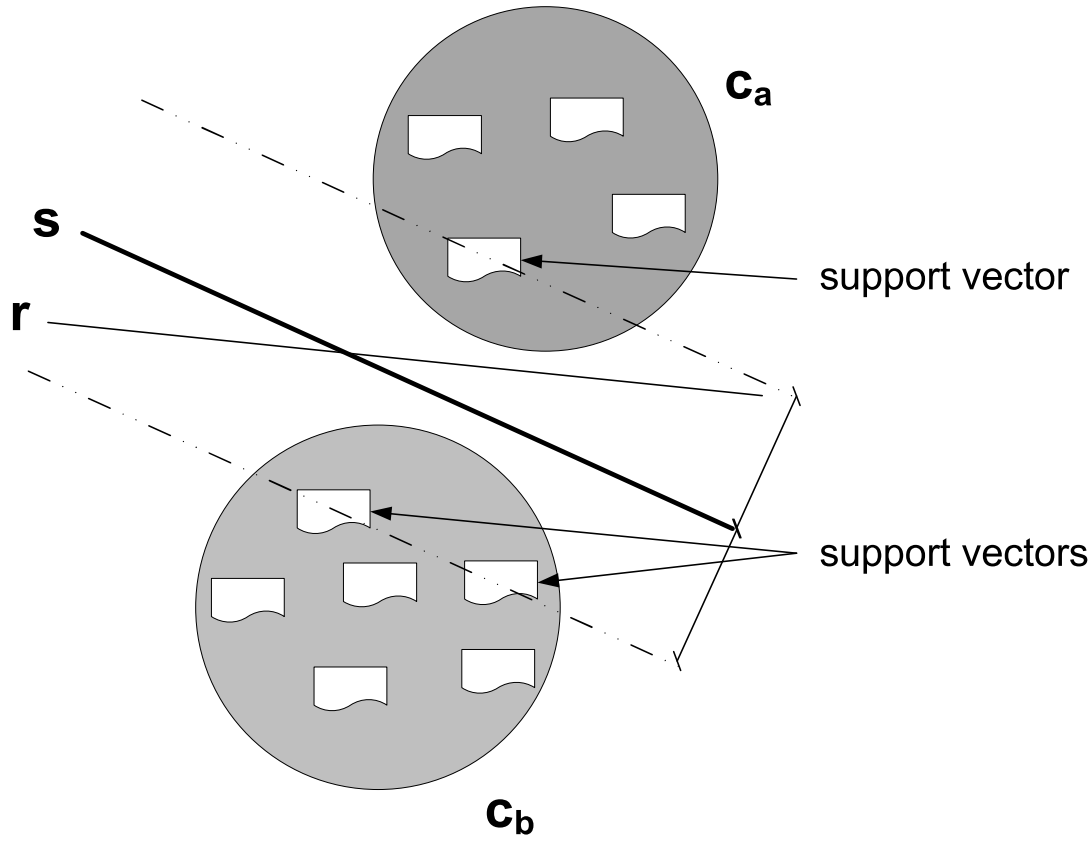
The SVM Classifier

SVM Basic Technique – Intuition

- **Support Vector Machines (SVMs)** constitute a vector space method for binary classification problems
- Consider a set of documents represented in a t-dimensional space
- The idea is to find a **decision surface (hyperplane)** that best separate the elements of two classes
- Then, a new document d_j can be classified by computing its position relative to the hyperplane

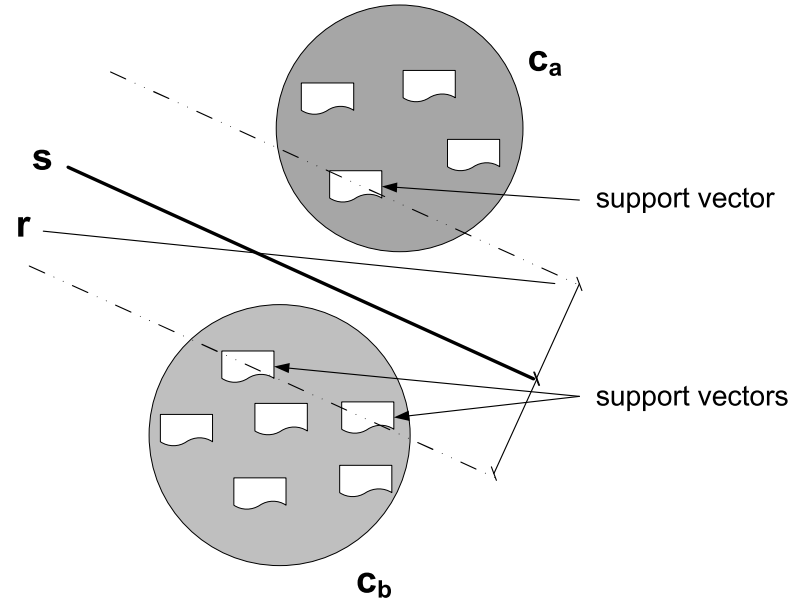
SVM Basic Technique – Intuition

- Consider a simple 2D example whose training data points are linearly separable, as illustrated below



SVM Basic Technique – Intuition

- Line s **maximizes** the distances to the closest documents of each class
- Then, s constitutes the best separating hyperplane, that we refer to the **decision hyperplane**
- In Figure, the parallel dashed lines delimit the region where to look for a solution
- We refer to them as the **delimiting hyperplanes**

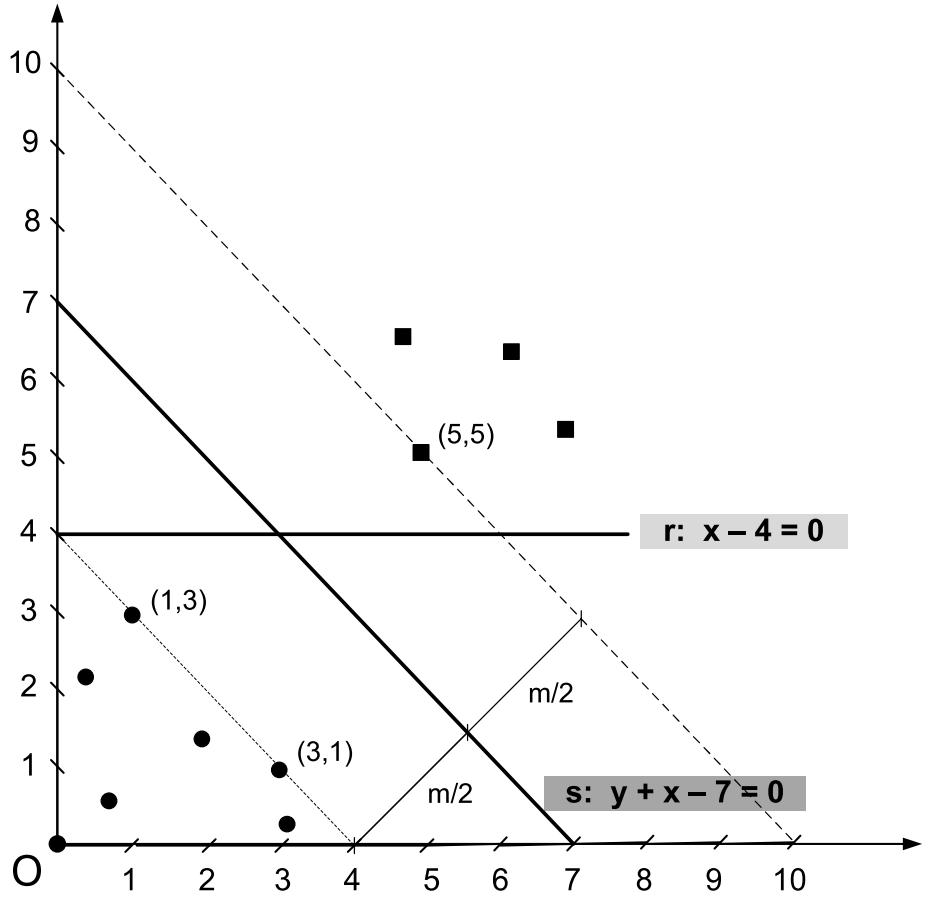


SVM Basic Technique – Intuition

- Lines that cross the delimiting hyperplanes are candidates to be selected as the decision hyperplane
- Lines that are parallel to the this space are the best candidates
- Documents that belong to the delimiting hyperplanes are called **support vectors**

SVM Basic Technique – Intuition

- Figure below illustrates our example in a 2-dimensional system of coordinates

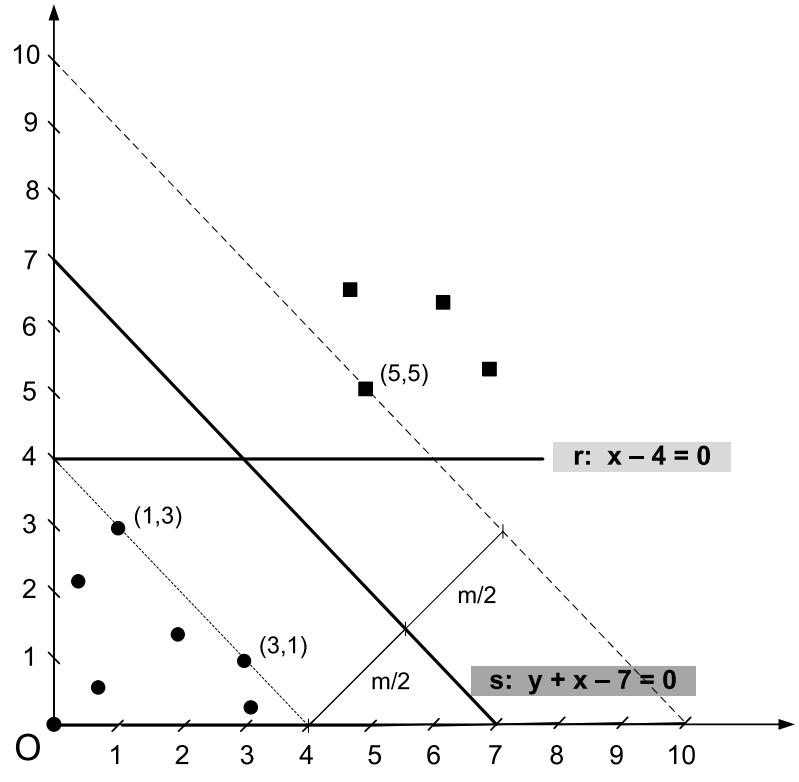


SVM Basic Technique – Intuition

- The SVM optimization problem can be stated as follows
- Let $\mathcal{H}_w$ be a hyperplane that separates all docs in class c_a from all docs in class c_b
- Further, Let
  - m_a be the distance of $\mathcal{H}_w$ to the closest document in class c_a
  - m_b be the distance of $\mathcal{H}_w$ to the closest document in class c_b
  - $m_a + m_b$ is called the **margin** m of the SVM
- The decision hyperplane $\mathcal{H}_w$ maximizes the margin m

SVM Basic Technique – Intuition

- In our example, the hyperplane $r : x - 4 = 0$ separates the documents in the two sets
- It has distances to the closest documents in either class, points (1,3) and (5,5), equal to 1
  - Thus, its margin m is 2
- The hyperplane $s : y + x - 7 = 0$ provides a margin equal to $3\sqrt{2}$, which is maximum for this case
- Then, the hyperplane s is the decision hyperplane



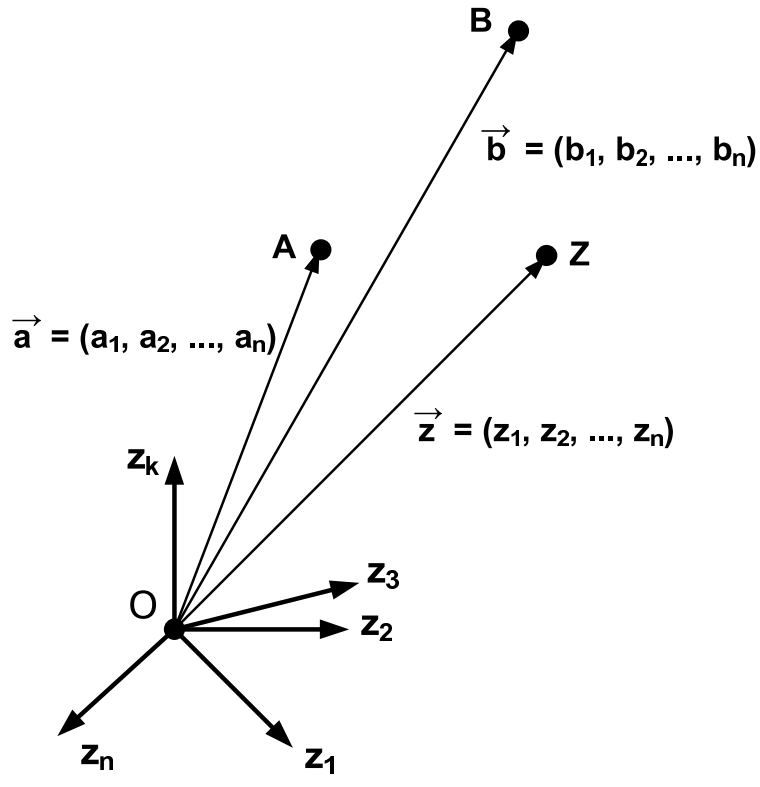
Lines and Hyperplanes in the $\mathcal{R}^n$

- Let $\mathcal{R}^n$ refer to an n-dimensional space with origin in $\mathcal{O}$
- In $\mathcal{R}^n$, a generic point Z is represented as a vector $\vec{z}$ given by

$$\vec{z} = (z_1, z_2, \ldots, z_n)$$

where z_i, $1 \leq i \leq n$, are real variables

- We adopt a similar notation to refer to specific fixed points such as A, B, H, P, and Q

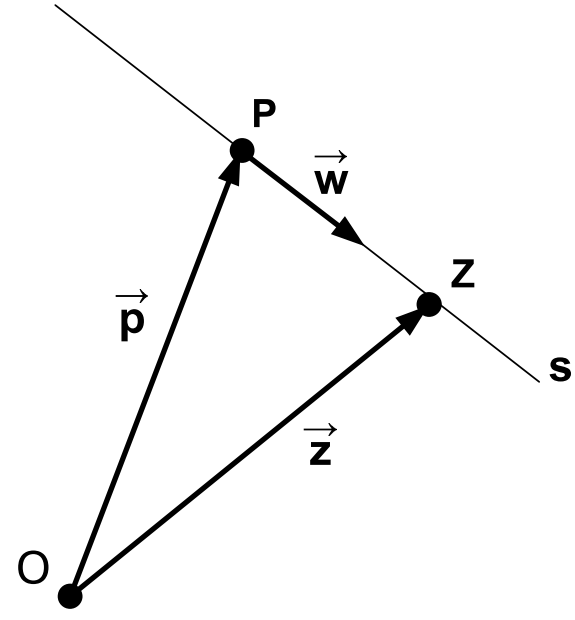


Lines and Hyperplanes in the $\mathcal{R}^n$

- Figure below illustrates a line s in the direction of a vector $\vec{w}$ that contains a given point P
- The parametric equation for this line can be written as

$$s : \vec{z} = t\vec{w} + \vec{p}$$

where $-\infty < t < +\infty$



Lines and Hyperplanes in the $\mathcal{R}^n$

- Figure below illustrates a hyperplane $\mathcal{H}_w$ that contains a point H and is perpendicular to a given vector $\vec{w}$

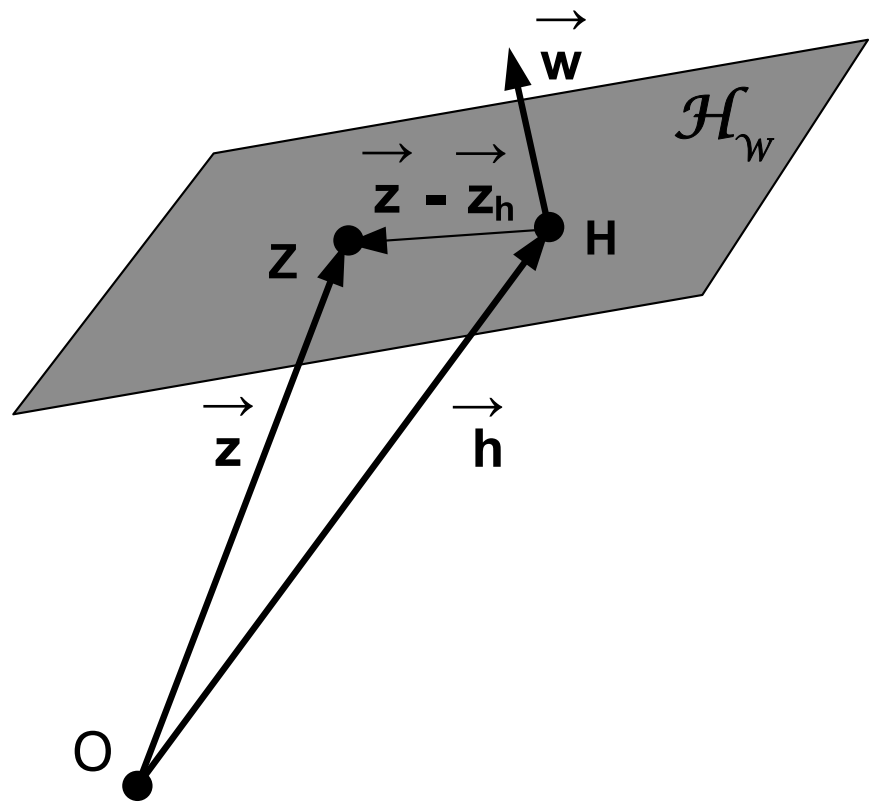
- The normal equation for this hyperplane is

$$\mathcal{H}_w : (\vec{z} - \vec{h})\vec{w} = 0$$

- This equation can be rewritten as

$$\mathcal{H}_w : \vec{z}\vec{w} + k = 0$$

where $\vec{w}$ and $k = -\vec{h}\vec{w}$ need to be determined

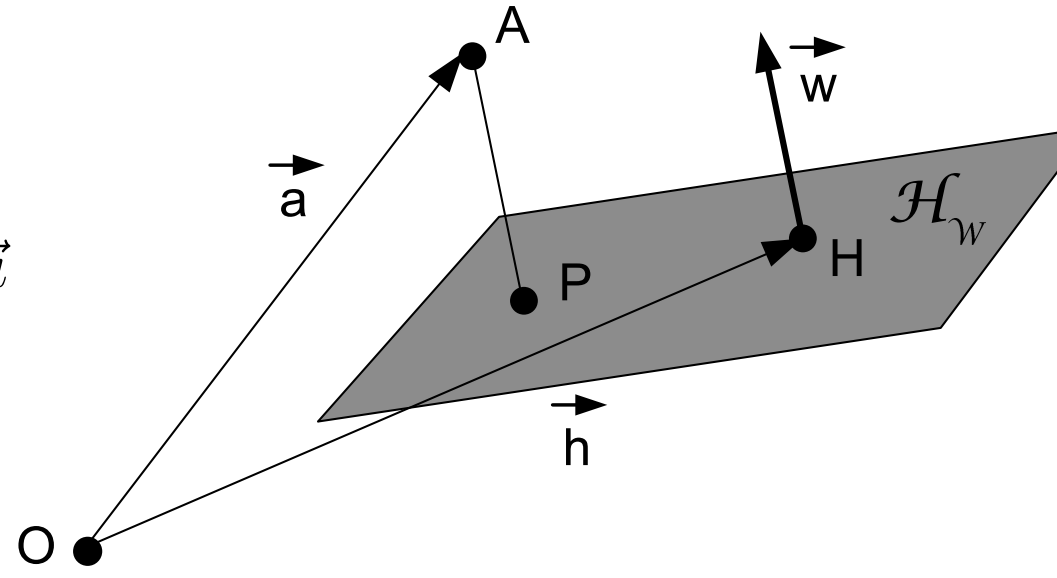


Lines and Hyperplanes in the $\mathcal{R}^n$

- Let P the projection of a point A on the hyperplane
- The distance of a point A to a hyperplane is given by the segment $\overline{AP}$
- The parametric equation of the line determined by points A and P is

$$line(\overline{AP}) : \vec{z} = t\vec{w} + \vec{a}$$

where $-\infty < t < +\infty$



Lines and Hyperplanes in the $\mathcal{R}^n$

- For the point P specifically, we have

$$\vec{p} = t_p \vec{w} + \vec{a}$$

where t_p is the value of t for point P

- Since $P \in \mathcal{H}_w$, we obtain

$$(t_p \vec{w} + \vec{a})\vec{w} + k = 0$$

- Solving for t_p,

$$t_p = -\frac{\vec{a}\vec{w} + k}{|\vec{w}|^2}$$

where $|\vec{w}|$ is the vector norm

Lines and Hyperplanes in the $\mathcal{R}^n$

- By substituting t_p back into Equation of the point P, we obtain

$$\vec{a} - \vec{p} = \frac{\vec{a}\vec{w} + k}{|\vec{w}|} \times \frac{\vec{w}}{|\vec{w}|}$$

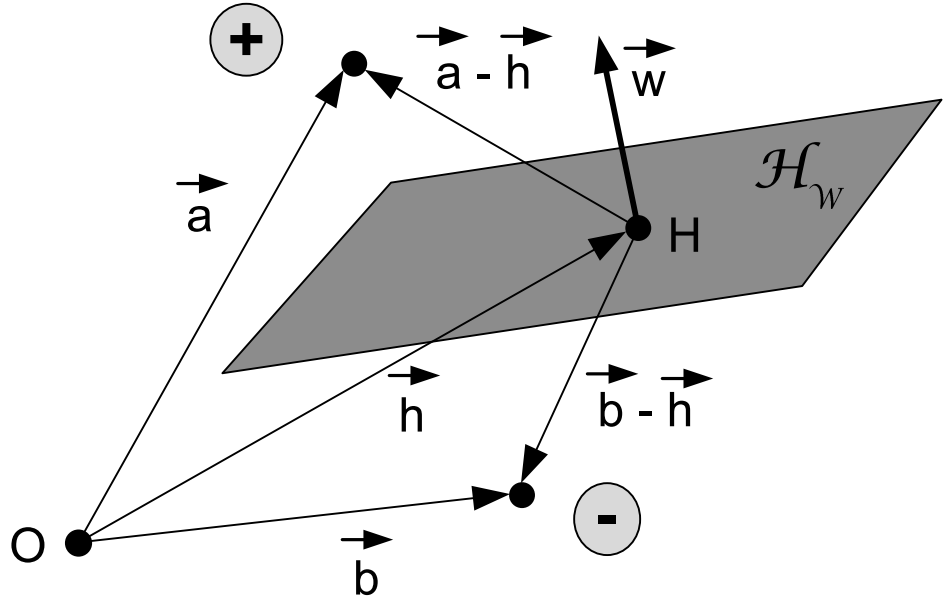
- Since $\vec{w}/|\vec{w}|$ is a unit vector in the direction of $\vec{w}$, we have

$$\overline{AP} = |\vec{a} - \vec{p}| = \frac{\vec{a}\vec{w} + k}{|\vec{w}|}$$

which is the distance of point A to hyperplane $\mathcal{H}_w$

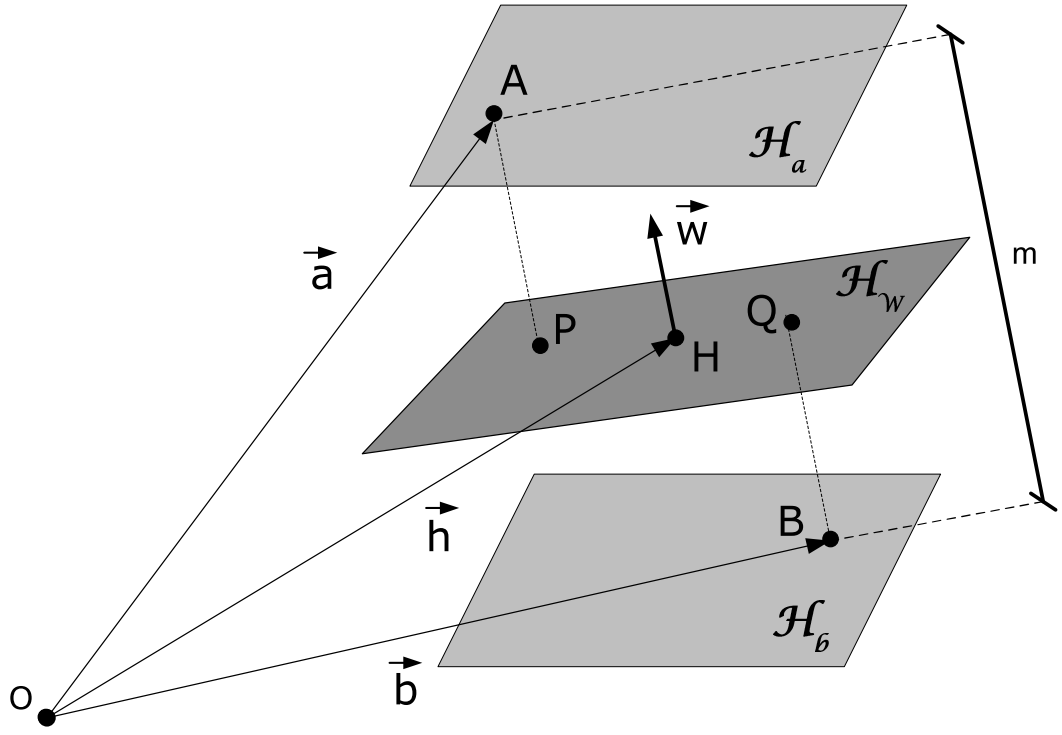
Lines and Hyperplanes in the $\mathcal{R}^n$

- Figure below illustrates how signs vary with regard to a hyperplane $\mathcal{H}_w$
  - The region above the hyperplane $\mathcal{H}_w$ is composed of points $\vec{z}$ that make $\vec{z}\vec{w} + k$ positive
  - The region below the hyperplane is composed of points that make $\vec{z}\vec{w} + k$ negative



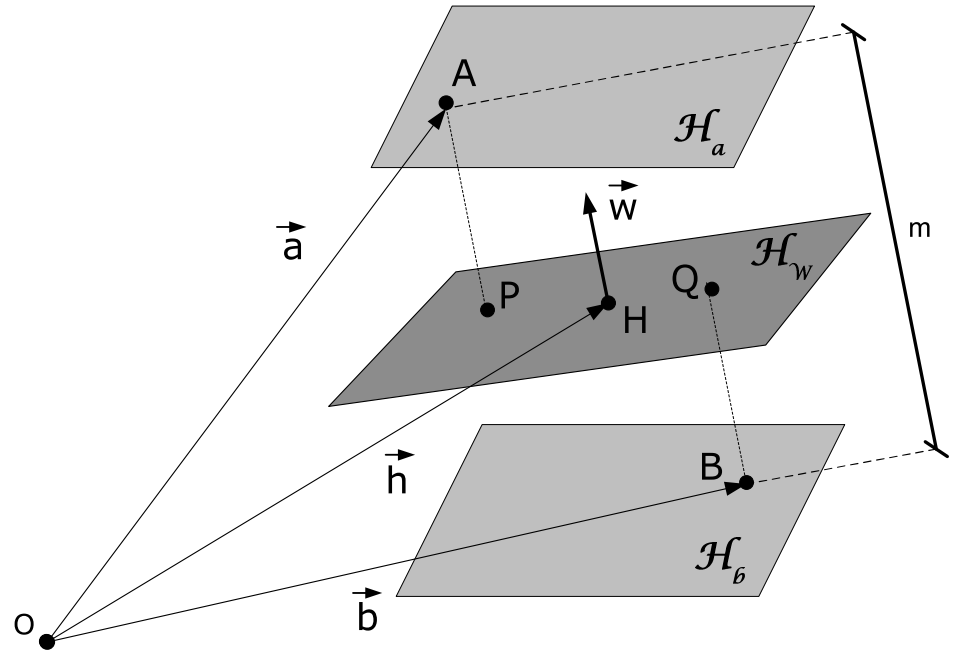
SVM Technique – Formalization

- **The SVM optimization problem:** given support vectors such as $\vec{a}$ and $\vec{b}$, find the hyperplane $\mathcal{H}_w$ that maximizes the margin m



SVM Technique – Formalization

- The origin of the coordinate system is point O
- The point A represent a doc from the class c_a and the point B a doc from the class c_b
- These points belong to the delimiting hyperplanes $\mathcal{H}_a$ and $\mathcal{H}_b$
- $\mathcal{H}_w$ is determined by a point H (represented by $\vec{h}$) and by a perpendicular vector $\vec{w}$
- Neither $\vec{h}$ nor $\vec{w}$ are known a priori



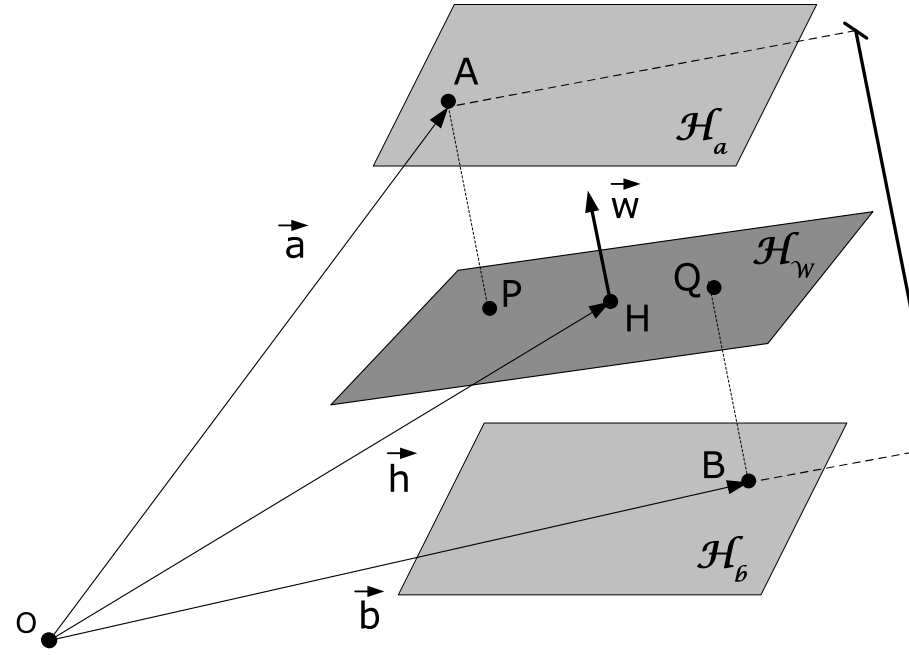
SVM Technique – Formalization

- Let P be the projection of point A on the hyperplane $\mathcal{H}_w$
- The distance of point A to the hyperplane is given by segment $\overline{AP}$, that is

$$\overline{AP} = \frac{\vec{a}\vec{w} + k}{|\vec{w}|}$$

- The distance of point B to the hyperplane is given by segment $\overline{BQ}$, that is

$$\overline{BQ} = -\frac{\vec{b}\vec{w} + k}{|\vec{w}|}$$



SVM Technique – Formalization

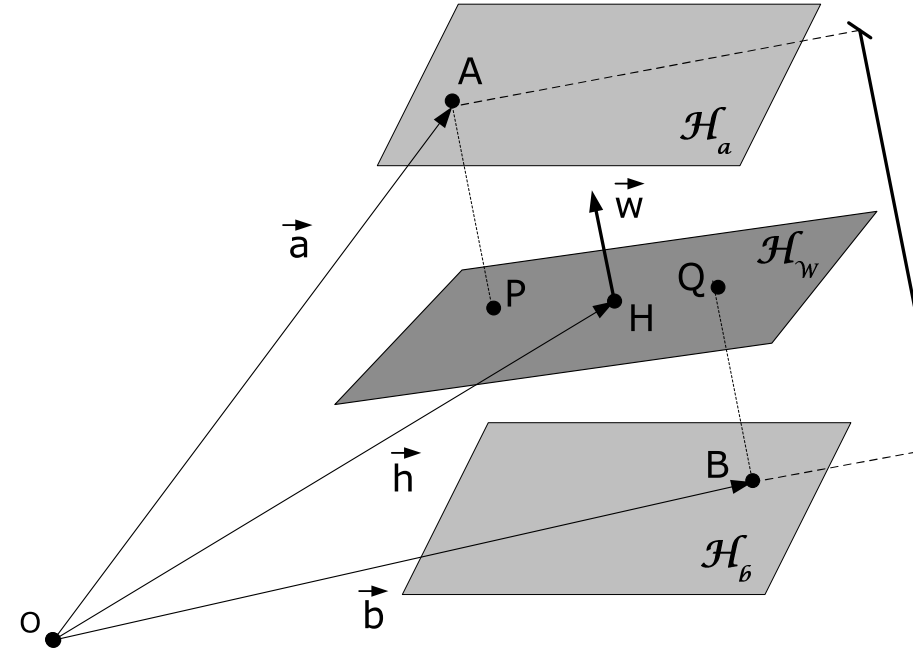
- From Figure below, the margin m of the SVM is given by

$$m = \overline{AP} + \overline{BQ}$$

and is independent of the size of $\vec{w}$

- That is, there are vectors $\vec{w}$ of varying sizes that maximize m
- To impose restrictions on $|\vec{w}|$, we can set

$$\begin{aligned} \vec{a}\vec{w} + k &= 1 \\ \vec{b}\vec{w} + k &= -1 \end{aligned}$$



SVM Technique – Formalization

- Notice that this also restricts the solution to hyperplanes that split the margin m in the middle
- Under these conditions, the expression for the margin m becomes

$$m = \frac{1}{|\vec{w}|} + \frac{1}{|\vec{w}|} = \frac{2}{|\vec{w}|}$$

SVM Technique – Formalization

- Let $\mathcal{T} = \{\ldots, [c_j, \vec{z}_j], [c_{j+1}, \vec{z}_{j+1}], \ldots\}$, be the training set
  - c_j is the class (either c_a or c_b) associated with point $\vec{z}_j$, i.e., with a document d_j
- Then,

  SVM Optimization Problem:

  maximize $m = 2/|\vec{w}|$
  subject to

  $\vec{w}\vec{z}_j + b \geq +1$ if $c_j = c_a$
  $\vec{w}\vec{z}_j + b \leq -1$ if $c_j = c_b$

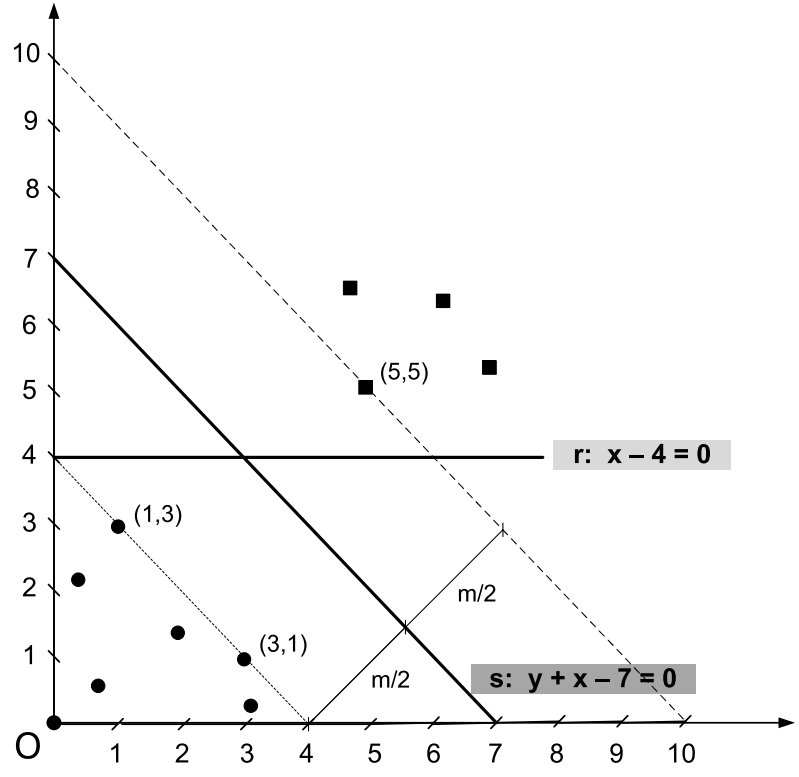
- The vectors that make the equation equal to either 1 or -1 are the support vectors

SVM Technique – Formalization

- Let us consider again the simple example in Figure below
- For that case, the optimization problem can be especified as:

  maximize $m = 2/|\vec{w}|$
  subject to

$$\vec{w} \cdot (5,5) + b = +1$$
$$\vec{w} \cdot (1,3) + b = -1$$



SVM Technique – Formalization

- If we represent vector $\vec{w}$ as (x, y) then $|\vec{w}| = \sqrt{x^2 + y^2}$
- The parameter m stands for the distance between the delimiting hyperplanes and is equal to $3\sqrt{2}$
- Thus,

$$\begin{aligned} 3\sqrt{2} &= 2/\sqrt{x^2 + y^2} \\ 5x + 5y + b &= +1 \\ x + 3y + b &= -1 \end{aligned}$$

which yields $b = -16/9$ or $b = -21/9$

SVM Technique – Formalization

- The value that maximizes $2/|\vec{w}|$ is $b = -21/9$, which yields $x = 1/3, y = 1/3$
- Thus, the equation of the decision hyperplane is given by

$$(1/3, 1/3) \cdot (x, y) + (-21/9) = 0$$

or

$$y + x - 7 = 0$$

Classification of Documents

- The classification of a doc d_j, represented as a vector $\vec{z}_j$, is achieved by applying the decision function

$$f(\vec{z}_j) = sign(\vec{w}\vec{z}_j + b)$$

- If the sign is positive then the document d_j belongs to class c_a; otherwise, it belongs to class c_b
- The SVM classifier might also enforce the margin to reduce classification errors
- In this case a new document d_j
  - is classified in class c_a only if $\vec{w}\vec{z}_j + b > 1$, and
  - is classified in class c_b only if $\vec{w}\vec{z}_j + b < -1$

SVM with Multiple Classes

- SVMs can only take binary decisions: a document belongs or not to a given class
- With multiple classes, strategies for reducing the multi-class problem to larger binary classification problems are used
- A natural way of doing so is to consider one binary classification problem per class

SVM with Multiple Classes

- To classify a new document d_j, we run the classification procedure for each class
- In each case, a distinct class c_p is paired against all others
- A classification decision is taken for document d_j
- To decide the class of d_j, we can take the classes that present the largest margins for the classification of $\vec{d}_j$

SVM with Multiple Classes

- Another possibility is to consider a binary classifier for each pair of classes c_p and c_q
- In this case:
  - All training documents of one class are considered as positive examples and
  - All documents from the other class are considered as negative examples

Non-Linearly Separable Cases

- SVM has no solutions when there is no hyperplane that separates the data points into two disjoint sets
  - This condition is known as **non-linearly separable case**
- In this case, a solution might be engineered by
  - **soft margin approach:** allowing the classifier to make a few mistakes, or
  - **kernel approach**: mapping the original data onto a higher dimensional space where the mapped data is linearly separable

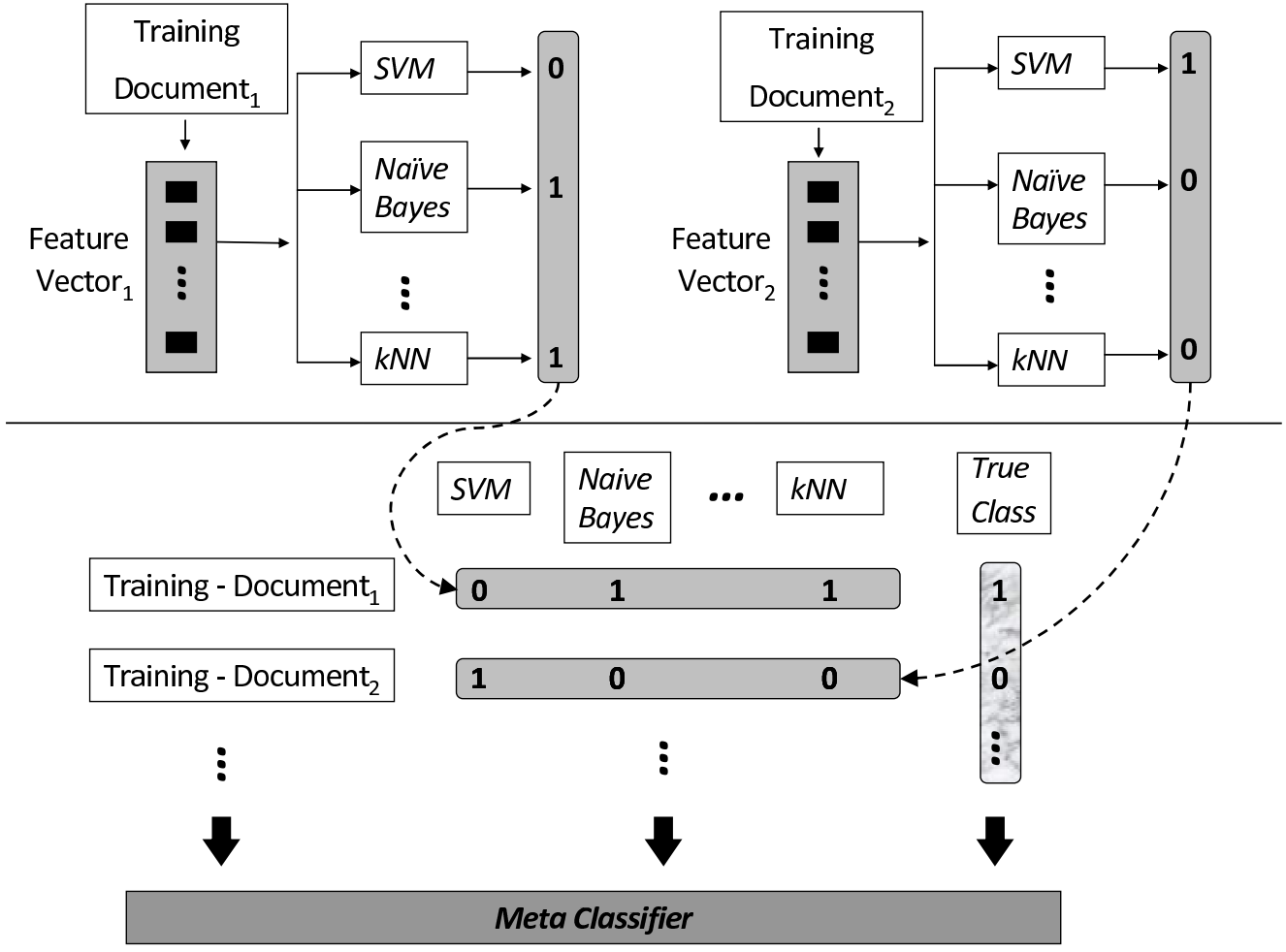
Ensemble Classifiers

Ensemble Classifiers

- An ensemble classifier combines the predictions of distinct classifiers to generate a new predictive score
- Ideally, this method produces results of higher precision than those yielded by the constituent classifiers
- We discuss two ensemble classification methods: **stacking** and **boosting**

Ensemble Classifiers

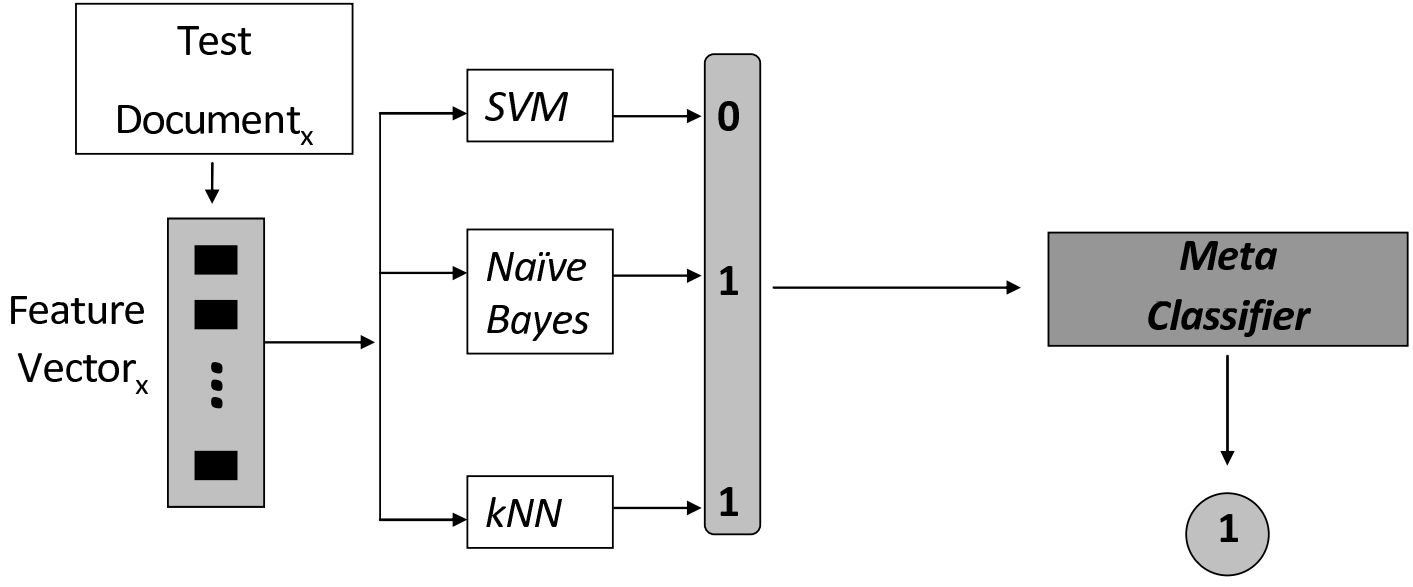
- Training phase of a stacking-based ensemble classifier



Ensemble Classifiers
Stacking-based Ensemble Classifiers

Stacking-based Classifiers

- The stacking method: to learn a function that combines the predictions of the individual classifiers



Stacking-based Classifiers

- With each document-class pair $[d_j, c_p]$ in the training set is associated the predictions made by distinct classifiers
- Instead of attempting to predict the correct class for a given document d_j, the classifier attempts to:
  - predict the base classifier that best predicts the class for d_j or
  - combine the predictions of the base classifiers to produce better results
- This method has the advantage that the errors of a base classifier can be counter-balanced by the hits of others

Ensemble Classifiers
Boosting-based Ensemble Classifiers

Boosting-based Classifiers

- Boosting: the classifiers to be combined are generated by several iterations of a **same learning technique**
- Focus on training documents that previous versions of the base classifier have mistakenly categorized
- At each interaction, each document in the training set is given a weight
- Weights of incorrectly classified documents are increased at each round
- After n rounds, the outputs of the trained classifiers are combined in a weighted sum, whose weights are the error estimates of each classifier

Boosting-based Classifiers

- A variation of the AdaBoost learning algorithm, originally proposed in Yoav Freund *et al*

AdaBoost

let $\mathcal{T} : \mathcal{D}_t \times \mathcal{C}$ be the training set function;

let N_t be the training set size and M be the number of iterations;

initialize the weight w_j of each document d_j as $w_j = \frac{1}{N_t}$;

for $k = 1$ to M {

learn the classifier function $\mathcal{F}_k$ from the training set;

estimate weighted error: $err_k = \sum_{d_j | d_j misclassified} w_j \;/\; \sum_{i=1}^{N_t} w_j$;

compute a classifier weight: $\alpha_k = \frac{1}{2} \times \log\left(\frac{1-err_k}{err_k}\right)$;

for all correctly classified examples e_j: $w_j \leftarrow w_j \times e^{-\alpha_k}$;

for all incorrectly classified examples e_j: $w_j \leftarrow w_j \times e^{\alpha_k}$;

normalize the weights w_j so that they sum up to 1;

}

Feature Selection or Dimensionality Reduction

Feature Selection

- A large feature space might render document classifiers impractical
- The classic solution to this problem is to select a subset of all features to represent the documents
- This step, that is called **feature selection**, contributes to
  - Reduce the dimensionality of the documents representation
  - Reduce **overfitting**

Term-Class Incidence Table

- The feature selection is dependent on statistics on the occurrence of terms inside documents and classes
- Let
  - $\mathcal{D}_t$ be the subset composed of all training documents
  - N_t be the number of documents in $\mathcal{D}_t$
  - t_i be the number of documents from $\mathcal{D}_t$ that contain the term k_i
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ be the set of all L classes
- Assume that a training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$ has been specified

Term-Class Incidence Table

- A term-class incidence table can be given by

Case	Docs in c_p	Docs not in c_p	Total
Docs that contain k_i	$n_{i,p}$	$n_i - n_{i,p}$	n_i
Docs that do not contain k_i	$n_p - n_{i,p}$	$N_t - n_i - (n_p - n_{i,p})$	$N_t - n_i$
All docs	n_p	$N_t - n_p$	N_t

- The number of documents that contain term k_i and are classified in class c_p is given by $n_{i,p}$
- The number of documents that contain term k_i but are not in class c_p is given by $n_i - n_{i,p}$
- n_p is the total number of training documents in class c_p

Term-Class Incidence Table

- $n_p - n_{i,p}$ is the number of documents from c_p that do not contain term k_i
- Given the term-class incidence table above, we can define various probabilities of interest, as follows
  - Probability that $k_i \in d_j$: $P(k_i) = \frac{n_i}{N_t}$
  - Probability that $k_i \notin d_j$: $P(\overline{k}_i) = \frac{N_t - n_i}{N_t}$
  - Probability that $d_j \in c_p$: $P(c_p) = \frac{n_p}{N_t}$
  - Probability that $d_j \notin c_p$: $P(\overline{c}_p) = \frac{N_t - n_p}{N_t}$
  - Probability that $k_i \in d_j$ and $d_j \in c_p$: $P(k_i, c_p) \frac{n_{i,p}}{N_t}$
  - Probability that $k_i \notin d_j$ and $d_j \in c_p$: $P(\overline{k}_i, c_p) = \frac{n_p - n_{i,p}}{N_t}$
  - Probability that $k_i \in d_j$ and $d_j \notin c_p$: $P(k_i, \overline{c}_p) = \frac{n_i - n_{i,p}}{N_t}$
  - Probability that $k_i \notin d_j$ and $d_j \notin c_p$: $P(\overline{k}_i, \overline{c}_p) = \frac{N_t - n_i - (n_p - n_{i,p})}{N_t}$

Feature Selection or Dimensionality Reduction

Term Document Frequency

Term Document Frequency

- Feature Selection by Term Document Frequency:
  - Let K_{th} be a threshold on term document frequencies
  - Then, all terms k_i for which $n_i \geq K_{th}$ are retained, all other terms are discarded
  - Documents representations are recomputed to consider only the terms retained
- Even if simple, this method allows considerably reducing the dimensionality of the space with no loss in effectiveness

Feature Selection or Dimensionality Reduction

Tf-Idf Weights

Tf-Idf Weights

- A term selection procedure that retains the terms of higher tf-idf weights in each document d_j
- Feature Selection by TF-IDF Weights:
  - Let $w_{i,j}$ refer to the tf-idf weight associated with the term-document pair $[k_i, d_j]$
  - Further, let K_{th} be a threshold on tf-idf weights
  - Then, all terms k_i for which $w_{i,j} \geq K_{th}$ are retained, all other terms are discarded
  - Documents representations are recomputed to consider only the terms retained
- Some experiments suggest that this feature selection allows reducing the dimensionality of the space by a factor of 10 with no loss in effectiveness

Feature Selection or Dimensionality Reduction

Mutual Information

Mutual Information

- **Mutual information** is a measure of the relative entropy between the distributions of two random variables
- If those variables are independent, we say that their mutual information is zero
  - In this case, knowledge of one of the variables does not allow infering anything about the other variable

Mutual Information

- Mutual information is expressed as

$$I(k_i, c_p) = \log \frac{P(k_i, c_p)}{P(k_i)P(c_p)} = \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}}$$

  across all classes

- That is,

$$\begin{aligned} MI(k_i, C) &= \sum_{p=1}^{L} P(c_p)\ I(k_i, c_p) \\ &= \sum_{p=1}^{L} \frac{n_p}{N_t}\ \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}} \end{aligned}$$

Mutual Information

- An alternative is to use the maximum term information over all classes, as follows:

$$
\begin{aligned}
I_{max}(k_i, C) &= max_{p=1}^{L}\ I(k_i, c_p) \\
&= max_{p=1}^{L}\ \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}}
\end{aligned}
$$

- Feature Selection by Entropy
  - Let K_{th} be a threshold on entropy
  - Then, all terms k_i for which $MI(k_i, C) \geq K_{th}$ are retained, all other terms are discarded
  - Documents representations are recomputed to consider only the terms retained

Feature Selection or Dimensionality Reduction

Information Gain

Information Gain

- Mutual information uses the probabilities associated with the occurrence of terms in documents
- **Information Gain** is a complementary metric, that also considers the probabilities associated with the absence of terms in documents
- It balances the effects of term/document occurrences with the effects of term/document absences

Information Gain

- The information gain $IG(k_i, \mathcal{C})$ of term k_i over the set $\mathcal{C}$ of all classes is defined as follows

$$IG(k_i, \mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|k_i) - H(\mathcal{C}|\neg k_i)$$

where

  - $H(\mathcal{C})$ is the entropy of the set of classes $\mathcal{C}$
  - $H(\mathcal{C}|k_i)$ and $H(\mathcal{C}|\neg k_i)$ are the conditional entropies of $\mathcal{C}$ in the presence and in the absence of term k_i

- In information theory terms, $IG(k_i, \mathcal{C})$ is a measure of the amount of knowledge gained about $\mathcal{C}$ due to the fact that k_i is known

Information Gain

- Using the term-class incidence table defined previously, and recalling the expression for entropy, we can write,

$$
\begin{aligned}
IG(k_i, \mathcal{C}) \quad = \quad & -\sum_{p=1}^{L} P(c_p) \log P(c_p) \\
& - \left(-\sum_{p=1}^{L} P(k_i, c_p) \log P(c_p | k_i) \right) \\
& - \left(-\sum_{p=1}^{L} P(\overline{k}_i, c_p) \log P(c_p | \overline{k}_i) \right)
\end{aligned}
$$

Information Gain

- Applying Bayes rule, this can rewritten as

$$IG(k_i, \mathcal{C}) = -\sum_{p=1}^{L} \left(P(c_p) \log P(c_p) - P(k_i, c_p) \log \frac{P(k_i, c_p)}{P(k_i)} - P(\overline{k}_i, c_p) \log \frac{P(\overline{k}_i, c_p)}{P(\overline{k}_i)} \right)$$

- By applying the probability definitions used in mutual information metric, we can write

$$IG(k_i, \mathcal{C}) = -\sum_{p=1}^{L} \left(\frac{n_p}{N_t} \log \left(\frac{n_p}{N_t} \right) - \frac{n_{i,p}}{N_t} \log \frac{n_{i,p}}{n_i} - \frac{n_p - n_{i,p}}{N_t} \log \frac{n_p - n_{i,p}}{N_t - n_i} \right)$$

Information Gain

- Feature Selection by Information Gain
  - Let K_{th} be a threshold on information gain
  - Then, all terms k_i for which $IG(k_i, \mathcal{C}) \geq K_{th}$ are retained, all other terms are discarded
  - Documents representations are recomputed to consider only the terms retained

Feature Selection or Dimensionality Reduction

Chi Square

Chi Square

- The chi square metric quantifies the lack of independence between term k_i and class c_p
- It is a statistical metric defined as follows

$$\chi^2(k_i, c_p) = \frac{N_t \left(P(k_i, c_p)P(\neg k_i, \neg c_p) - P(k_i, \neg c_p)P(\neg k_i, c_p)\right)^2}{P(k_i)\ P(\neg k_i)\ P(c_p)\ P(\neg c_p)}$$

- Using the probabilities previously defined, we can write

$$\begin{aligned}\chi^2(k_i, c_p) &= \frac{N_t\ \left(n_{i,p}\ (N_t - n_i - n_p + n_{i,p}) - (n_i - n_{i,p})\ (n_p - n_{i,p})\right)^2}{n_p\ (N_t - n_p)\ n_i\ (N_t - n_i)} \\ &= \frac{N_t\ \left(N_t n_{i,p} - n_p n_i\right)^2}{n_p n_i (N_t - n_p)(N_t - n_i)}\end{aligned}$$

Chi Square

- To apply feature selection to term k_i, we compute either average or max term values of chi square as follows

$$\chi^2_{avg}(k_i) = \sum_{p=1}^{L} P(c_p)\ \chi^2(k_i, c_p)$$

$$\chi^2_{max}(k_i) = max^{L}_{p=1}\ \chi^2(k_i, c_p)$$

- Feature Selection by Chi Square
  - Let K_{th} be a threshold on chi square
  - Then, all terms k_i for which $\chi^2_{avg}(k_i) \geq K_{th}$ are retained, all other terms are discarded
  - Documents representations are recomputed to consider only the terms retained

Evaluation Metrics

Evaluation Metrics

- Evaluation is a very important step in the development of any text classification method
- Without proper evaluation, there is no way to determine how good is a newly proposed text classifier
- That is, evaluation is a key step to *validate* a newly proposed classification method
- Here, we describe some of the most used metrics to assess the quality of single label text classifiers
- We start defining a contingency table, that will be to describe the most used evaluation metrics

Contingency Table

- Let
  - $\mathcal{D}$ be a collection of documents
  - $\mathcal{D}_t$ be the subset composed of training documents
  - N_t be the number of documents in $\mathcal{D}_t$
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ be the set of all L classes
- Assume that have been specified
  - A training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$
  - A text classifier function $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$
- Further, let
  - n_t be the number of docs from the training set $\mathcal{D}_t$ in class c_p
  - n_f be the number of docs from the training set assigned to class c_p by the classifier

Contingency Table

- Consider the application of the classifier to all documents in the training set
- The contingency table is given by

$Case$	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	$Total$
$\mathcal{F}(d_j, c_p) = 1$	$n_{f,t}$	$n_f - n_{f,t}$	n_f
$\mathcal{F}(d_j, c_p) = 0$	$n_t - n_{f,t}$	$N_t - n_f - n_t + n_{f,t}$	$N_t - n_f$
$All\ docs$	n_t	$N_t - n_t$	N_t

  - where $n_{f,t}$ is the number of docs that both the training and classifier functions assigned to class c_p
  - The number of training docs in class c_p that were miss-classified by the classifier is given by $n_t - n_{f,t}$
  - The remaining quantities are calculated analogously

Evaluation Metrics
Contingency Table

Accuracy and Error

- The accuracy and error metrics are defined relative to a given class c_p, as follows

$$Acc(c_p) = \frac{n_{f,t} + (N_t - n_f - n_t + n_{f,t})}{N_t}$$

$$Err(c_p) = \frac{(n_f - n_{f,t}) + (n_t - n_{f,t})}{N_t}$$

- Notice that necessarily

$$Acc(c_p) + Err(c_p) = 1$$

Accuracy and Error

- Accuracy and error, despite their common use, have disadvantages
- Consider, for instance, a binary classification problem with only two categories c_p and c_r
- Assume that out of 1,000 documents there are only 20 in class c_p
- Then, a classifier that assumes that all docs are not in class c_p has an accuracy of 98% and an error of just 2%
- These values suggest that we have a very good classifier, but this is not the case

Accuracy and Error

- Consider now a second classifier that correctly predicts 50% of the documents in c_p, as illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- In this case, accuracy and error are given by

$$Acc(c_p) = \frac{10 + 980}{1,000} = 99\%$$

$$Err(c_p) = \frac{10 + 0}{1,000} = 1\%$$

Accuracy and Error

- This classifier is much better than one that guesses that all documents are not in class c_p
- However, its accuracy is just 1% better (it increased from 98% to 99%)
- This suggests that the two classifiers are almost equivalent, which is not the case.

Evaluation Metrics
Precision and Recall

Precision and Recall

- Precision and recall in text classification are variants of the precision and recall metrics in IR
- Precision P and recall R are computed relative to a given class c_p, as follows

$$P(c_p) = \frac{n_{f,t}}{n_f} \qquad R(c_p) = \frac{n_{f,t}}{n_t}$$

- Precision is the fraction of all docs assigned to class c_p by the classifier that really belong to class c_p
- Recall is the fraction of all docs that belong to class c_p that were correctly assigned to class c_p

Precision and Recall

- Consider again the the classifier illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- Precision and recall figures are given by

$$P(c_p) = \frac{10}{10} = 100\%$$

$$R(c_p) = \frac{10}{20} = 50\%$$

- That is, the classifier has precision of 100% and recall of 50% for class c_p

Precision and Recall

- Precision and recall defined in this way are computed for every category in set $\mathcal{C}$
- This yields a great number of values, making the tasks of comparing and evaluating algorithms more difficult
- It is often convenient to combine precision and recall into a single quality measure
- One of the most commonly used such measures is the *F-measure*, which we discuss now

Evaluation Metrics
F-measure

F-measure

- The F-measure combines precision and recall values
- It allows for the assignment of different weights to each of these measures
- It is defined as follows:

$$F_\alpha(c_p) = \frac{(\alpha^2 + 1)P(c_p)R(c_p)}{\alpha^2 P(c_p) + R(c_p)}$$

  where α defines the relative importance of precision and recall
- When $\alpha = 0$, only precision is considered. When $\alpha = \infty$, only recall is considered
- When $\alpha = 0.5$, recall is half as important as precision, and so on

F-measure

- The most used form of the *F-measure* is obtained by assigning equal weights to precision and recall
- This is accomplished by making $\alpha = 1$, and is is called the F_1-measure:

$$F_1(c_p) = \frac{2P(c_p)R(c_p)}{P(c_p) + R(c_p)}$$

F-measure

- Consider again the the classifier illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- For this example, we write

$$F_1(c_p) = \frac{2 * 1 * 0.5}{1 + 0.5} \sim 67\%$$

Evaluation Metrics

F_1 Macro and Micro Averages

F_1 Macro and Micro Averages

- It is also common to derive a unique F_1 value for a classifier
- This is accomplished computing the average of F_1 values across all individual categories
- There are two average functions that are considered in the literature:
  - Micro-average $F1$, or $micF_1$
  - Macro-average F_1, or $macF_1$

F_1 Macro and Micro Averages

- Macro-average F_1 is computed as

$$macF_1 = \frac{\sum_{p=1}^{|\mathcal{C}|} F_1(c_p)}{|\mathcal{C}|}$$

- Thus, macro-average F_1 simply averages F_1 across all categories

F_1 Macro and Micro Averages

- To compute micro-average F_1 we consider recall and precision figures over all categories, as follows

$$P = \frac{\sum_{c_p \in \mathcal{C}} n_{f,t}}{\sum_{c_p \in \mathcal{C}} n_f}$$

$$R = \frac{\sum_{c_p \in \mathcal{C}} n_{f,t}}{\sum_{c_p \in \mathcal{C}} n_t}$$

- Then, micro-average F_1 can be computed by

$$micF_1 = \frac{2PR}{P + R}$$

F_1 Macro and Micro Averages

- In micro-average F_1, every single document is given the same importance
- In macro-average F_1, every single category is given the same importance
- Macro-average F_1 captures better the ability of the classifier to perform well for many classes
- This becomes important whenever the distribution of classes is very skewed
- In this case, both average metrics should be considered

Evaluation Metrics
Cross-Validation

Cross-Validation

- Cross-validation has become a standard method to guarantee the statistical validation of the results
- It consists of building k different classifiers: $\Psi_1, \Psi_2, \ldots, \Psi_k$
- The classifiers are built by dividing the training set $\mathcal{D}_t$ into k disjoint sets or folds of sizes

$$N_{t1}, N_{t2}, \ldots, N_{tk}$$

Cross-Validation

- The classifier Ψ_i uses the training set minus the ith fold for tunning and run its test on the ith fold
- Each classifier is evaluated independently using precision-recall or F_1 figures
- The cross-validation is done by computing the average of the k measures
- The most commonly adopted value of k is 10, in which case the method is called **ten-fold cross-validation**

Evaluation Metrics
Standard Collections

Standard Collections

- Several standard benchmark collections are available for experimenting classification techniques
- In the immediately following, we present some of the most used benchmark collections

Standard Collections

- **Reuters-21578**
  - It is the most widely used collection in the classification experiments
  - It is constituted of news articles from Reuters for the 1987 year
  - The collection is classified under several categories related to economics (e.g., acquisitions, earnings, etc)
  - It contains 9,603 documents for training and 3,299 for testing, with 90 categories co-occuring in both training and test
  - Class proportions range from 1,88% to 29,96% in the training set and from 1,7% to 32,95% in the testing set

Standard Collections

- **Reuters Corpus Volume 1 (RCV1) and Volume 2 (RCV2)**
  - The RCV1 is another collection of news stories recently released by Reuters
  - It contains approximately 800,00 documents organized in 103 topical categories
  - It is expected to substitute the previous Reuters-21578 collection in text classification experiments
  - RCV2 is a modified version of the original released collection, in which some corrections were made

Standard Collections

- **OHSUMED**
  - OHSUMED is another popular collection used for testing text classification algorithms
  - It is a subset of the Medline digital library, containing medical documents (title or title + abstract)
  - There are 23 classes corresponding to MesH diseases used to index the documents

Standard Collections

- **20 NewsGroups**
  - A third largely used collection is 20 Newsgroups
  - This is a collection of approximately 20,000 messages posted to Usenet newsgroups, partitioned (nearly) evenly across 20 different newsgroups
  - The categories are the newsgroups themselves

Standard Collections

- Other collections reported in the text classification literature
  - WebKB hypertext collection
  - ACM-DL
  - a subset of the ACM Digital Library
  - samples of Web Directories such as Yahoo and ODP

Organizing the Classes – Taxonomies

Taxonomies

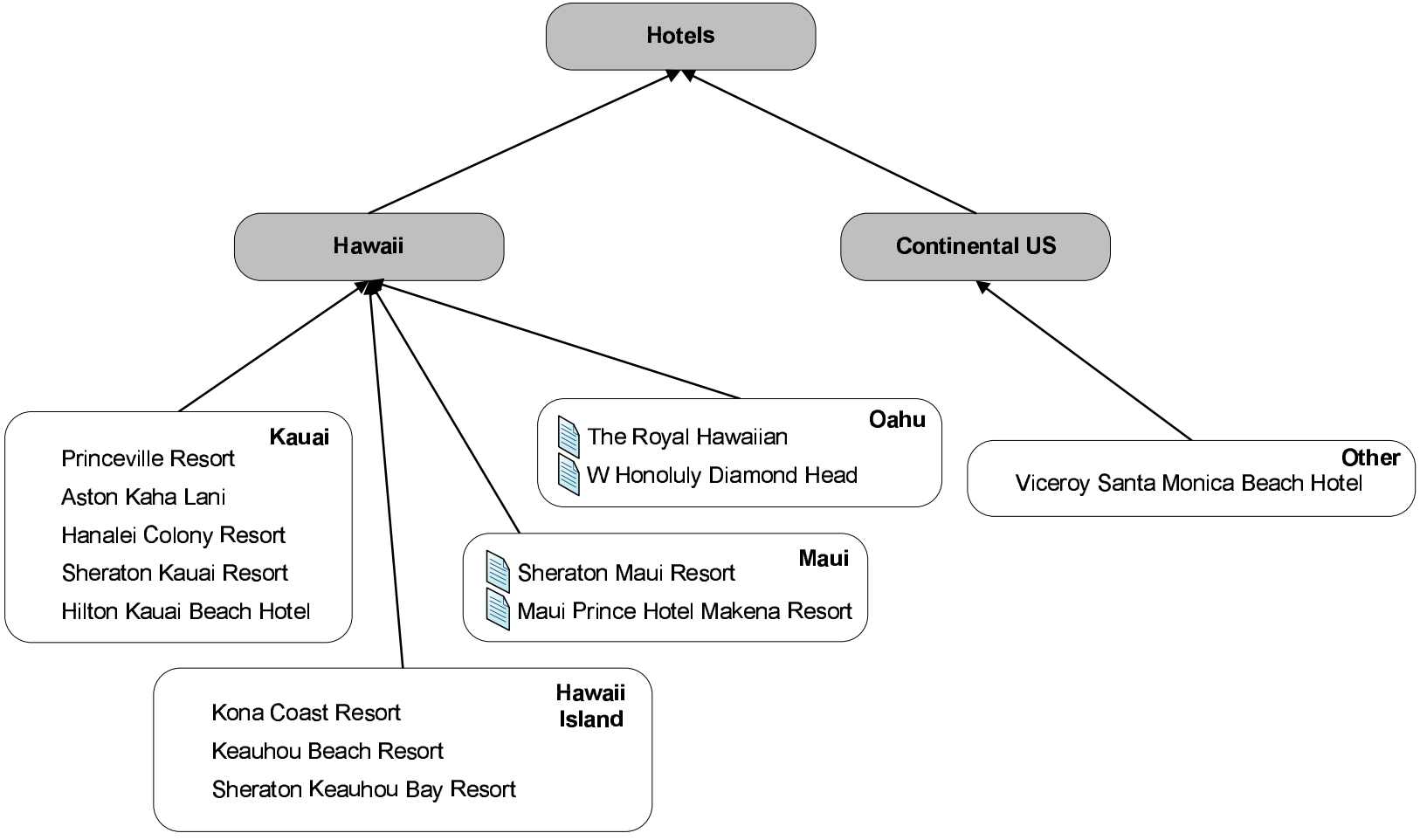
- Labeling provides information on the semantics of each class
- However, no organization of the classes is provided
- Lack of organization of the classes imposes restrictions to comprehension and reasoning
- Among all sorts of organization, the most appealing one is the **hierarchical organization**
- Hierarchies allow us to reason in terms of more generic concepts
- They also provide for specialization which allows breaking up a larger set of entities into subsets

Taxonomies

- We can organize the classes hierarchically using specialization, generalization, and sibling relations
- Classes organized hierarchically in this fashion compose a **taxonomy**
- In this case, the relations among the classes can be used to further fine tune the classifier
- Taxonomies make more sense when built for a specific domain of knowledge

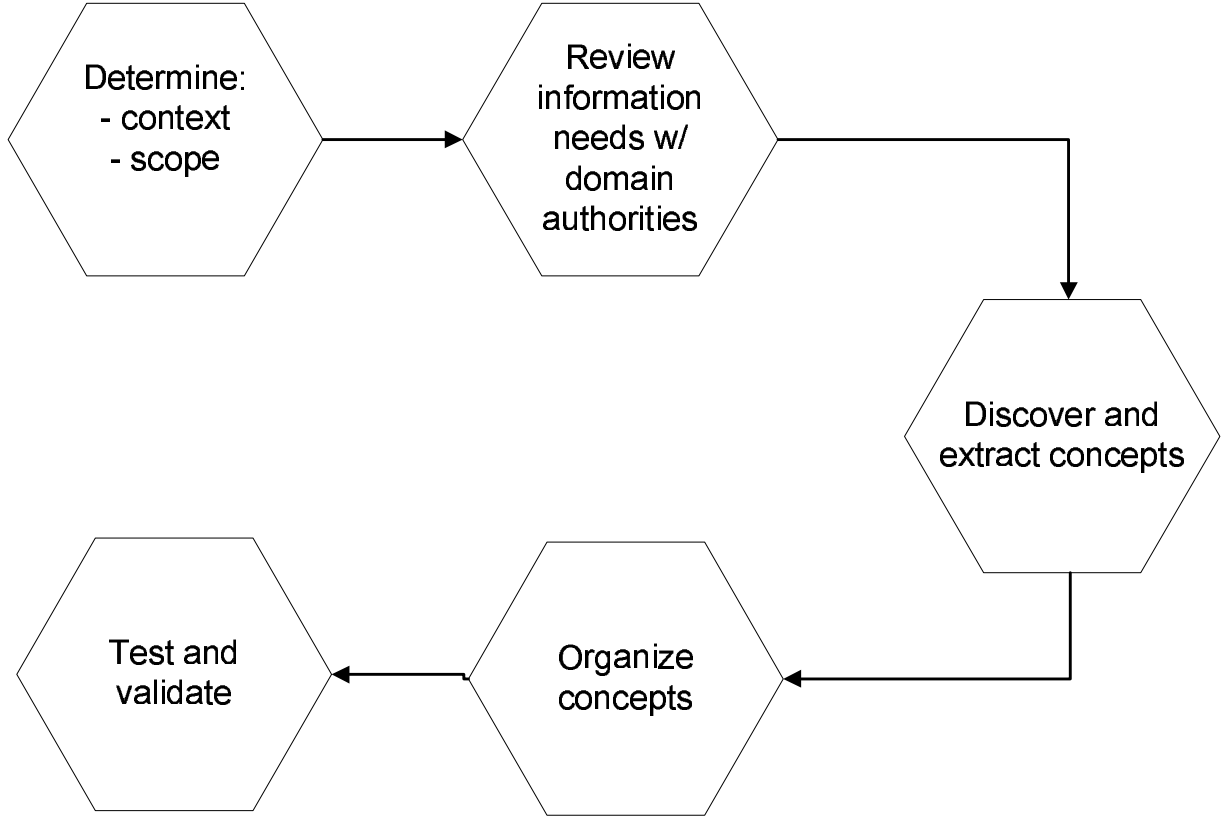
Taxonomies

- Organization of Web pages of hotels in Hawaii in a geo-referenced taxonomy



Taxonomies

- Usually, taxonomies are built **manually** or **semi-automatically** using complex procedures
- The process of building a taxonomy:



Taxonomies

- Manual taxonomies tend to be of superior quality, and better reflect the information needs of the users
- The automatic construction of taxonomies is one area of technology that needs more research and development
- Once a taxonomy has been built, the documents of the collection can be classified according to its concepts
- This can be accomplished manually or automatically
- Automatic classification algorithms are advanced enough to work well in practice